2

FINAL REPORT

Optical Drifting Buoy Deployments

FINAL REPORT

Title: Optical Drifting Buoy Deployments

To: Scientific Officer
Office of Naval Research
Code 252A GR
Ballston Tower One
800 North Quincy Street
Arlington, VA 22217-5660

Attn: Dr. Steve Ackleson

Contract No: (N00014-93-0211)

From: Satlantic Inc.
Pier 9, Richmond Terminals
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8
Telephone: 902-492-4780
Fax: 902-492-4781

Date: 31 December, 1993

Signatures:

Dr. Marlon R. Lewis/President

Accesion For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

1.0 Scope.

This document describes the construction, deployment, and initial data processing of four optical drifting buoys. The buoys, which measure spectral water-leaving radiance and downward irradiance, were successfully deployed in October/November, 1993 in the Equatorial Pacific in collaboration with the iron-seeding experiment IRONEX.

2.0. Optical Drifting Buoys.

The Ocean Color Monitor (OCM) optical drifting buoy represents a novel means to autonomously acquire, and retrieve, data on upwelling spectral radiance ("ocean color") and surface downwelling irradiance from remote ocean locations. The buoys are designed to be either ship, or aircraft, launched. Data is automatically telemetered in near to real time via CLS/ARGOS and Internet to Dalhousie computers. The buoys also measure sea-surface temperature.

Further details of the buoy construction, optical characterization, calibration and data reporting can be found in Appendix A, Ocean Color Monitor User's Manual.

3.0. Deployment and Operations

The buoys were packed at the factory, and shipped to Miami where they were loaded on the Columbus Iselin to take to the deployment site. Instructions were provided to the ships crew on deployment techniques.

The buoys were deployed in association with the joint ONR/NASA/NSR iron enrichment study south of the Galapagos Islands in October/November, 1993 (IRONEX). An area approximately 100km^2 was fertilized with an iron sulphate solution to evaluate the effect on the phytoplankton population.

After the patch was inoculated, the first buoy was deployed at the nominal patch center at 90°W 5°S on 28 October, 1993. The next two subsequent buoys were deployed two days and 6 days later respectively, near where the patch center was thought to be. The final buoy was deployed later during a survey of the Galapagos Island plume region near the Equator.

All buoys turned on and began transmitting soon after deployment.

4.0 Data Processing

The raw ARGOS telemetry data (Level 0 data; file extension *.ARG) is automatically posted once per day to Dalhousie computers via Internet. This hexidecimal data is processed to a Level 1 data set which consists of earth-located, time-stamped (time of transmission) calibrated radiances, irradiance, and temperature (file extension *.DAT).

The Level 1 data is subjected to further processing which does a number of operations. First, the time is corrected to the midpoint of the hourly averaged data in GMT. Second, the calibrated radiance data is used to estimate a spectral attenuation coefficient to propagate the radiance to and through the sea-surface. Using the downwelling irradiance at 490 nm, and a spectral atmospheric transmission model, each radiance waveband is normalized to the ratio of the spectral irradiance at the sea-surface relative to the extraterrestrial atmospheric irradiance to produce normalized, water-leaving radiances. These values represent the expected water-leaving radiances with the sun at zenith and with no intervening atmosphere; it provides a means to compare data from different illumination conditions as well as a direct comparison with similar quantities computed from satellite (SeaWiFS) observations.

The data often has transmission errors (see Users Manual), which to first order are corrected at this stage. Correction methods look for drops in battery voltage, and anomalous temperature readings. Duplicate transmissions are also removed at this step. Some hand processing is usually required to remove bad data points not detected by the processing program.

Finally, a first order pigment concentration is computed from the normalized water-leaving radiances using the global CZCS algorithm. For low irradiance levels (i.e. night-time), the last good pigment estimation is carried until sufficient irradiance is available to provide good radiance measurements.

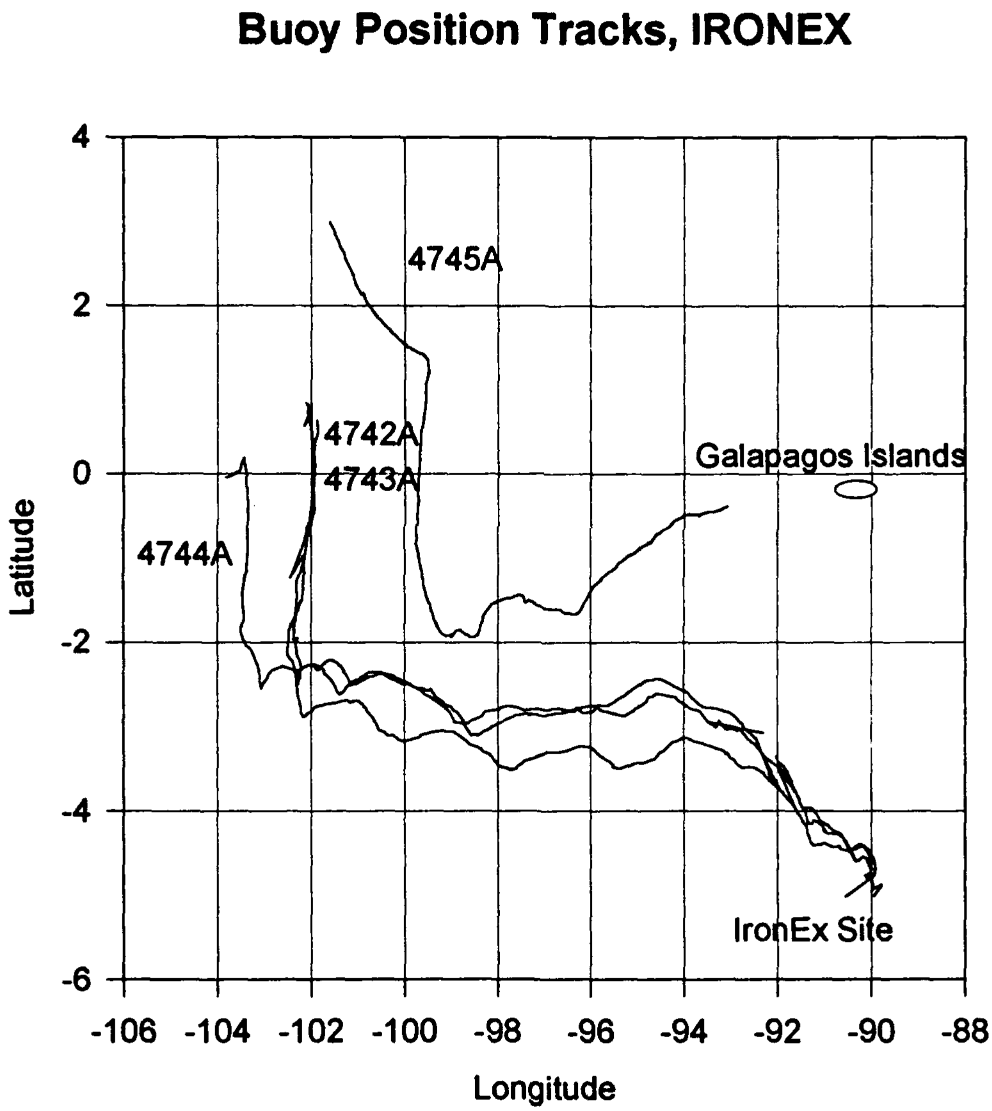
The result of these operations is a Level 2 data set (file extension *.PRO). These data can be further processed if desired into a variety of averaged or experimental products.

5.0. Initial Results and Discussion

As of the 28th of December, 1993, all buoys continued to transmit data. Buoys drifted initially in a northeasterly direction, then almost due north. Buoys 4742 and 4743 were deployed very close to each other, and remained close throughout

the past two months. As of 28 December, they were within 30 km of each other, despite the 2000 km they have translated since deployment. All buoys showed a high degree of coherence in their relative velocities and direction of drift. Figure 1 shows the buoy tracks until 22 December, 1993.

Figure 1. Optical Drifting Buoy Tracks.



5.1. Sea-surrface temperature. The data on sea-surface temperature shows both the long term variations as the buoys transited over their range, and also show a clear diurnal SST variation. The SST plots for the four buoys on a consistent time axis are shown in Figures 2-5 below.

Figure 2. Sea-surface temperature variations, Buoy 4745.

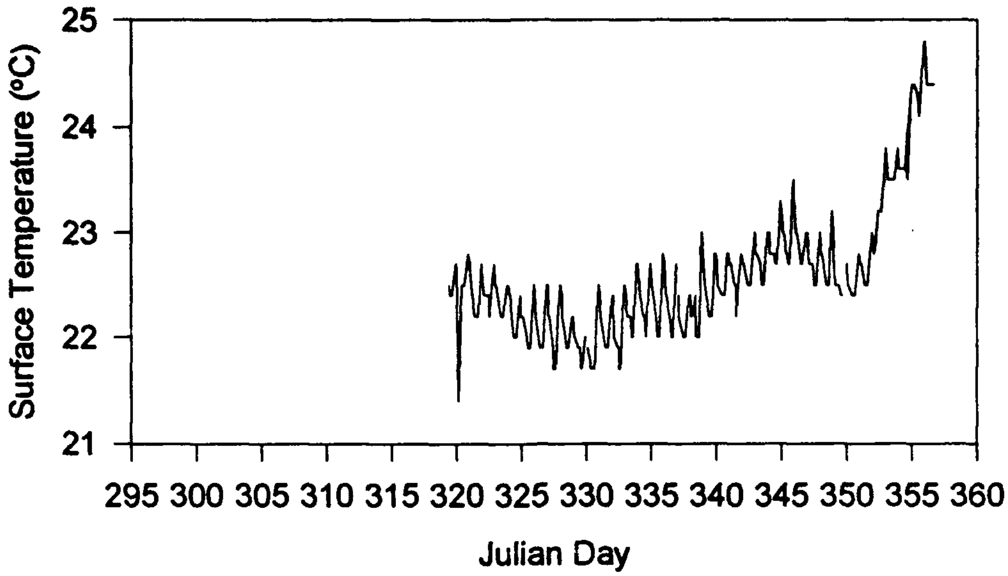


Figure 3. Sea-surface temperature variations, Buoy 4744.

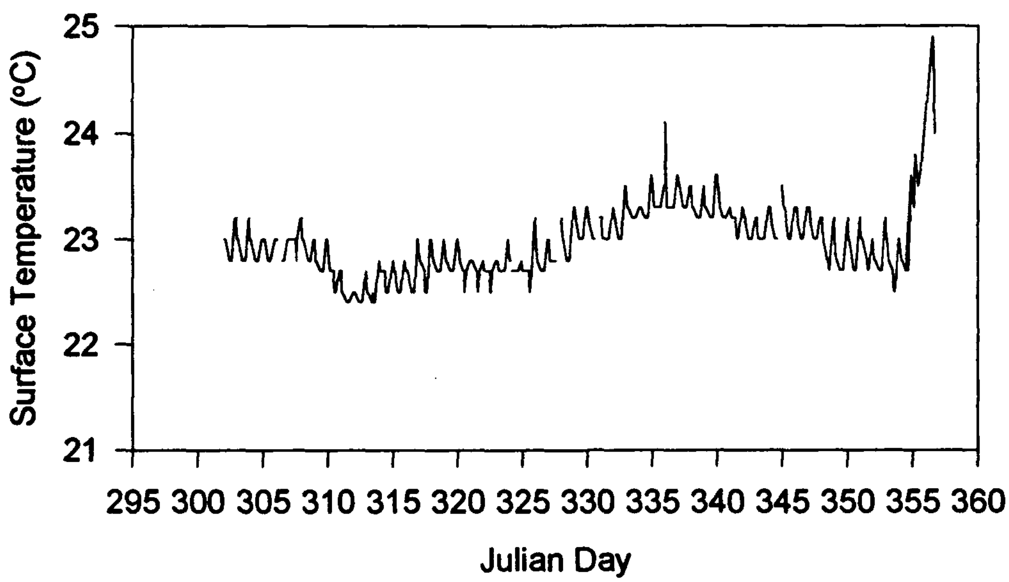


Figure 4. Sea-surface temperature variations, Buoy 4743.

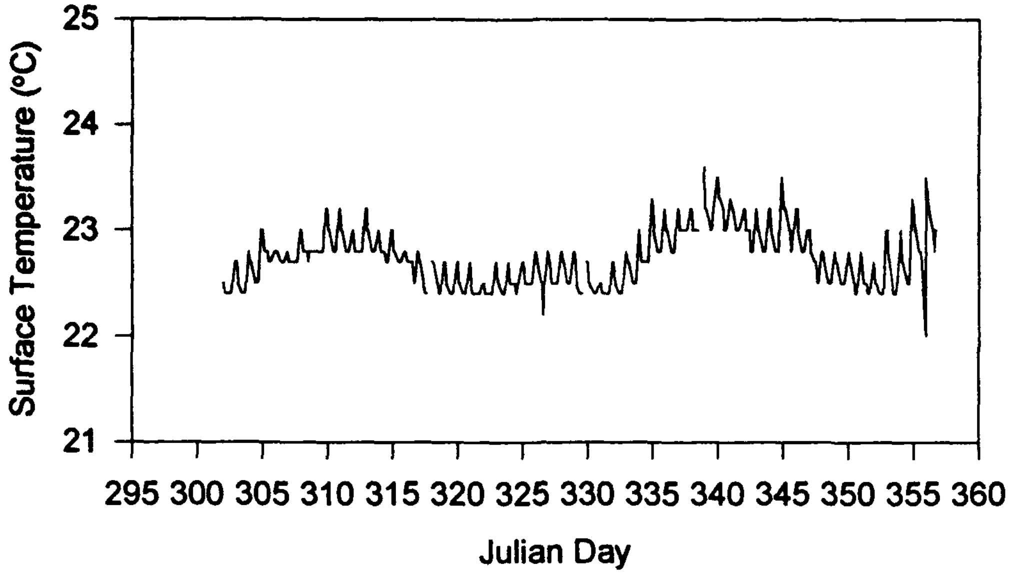
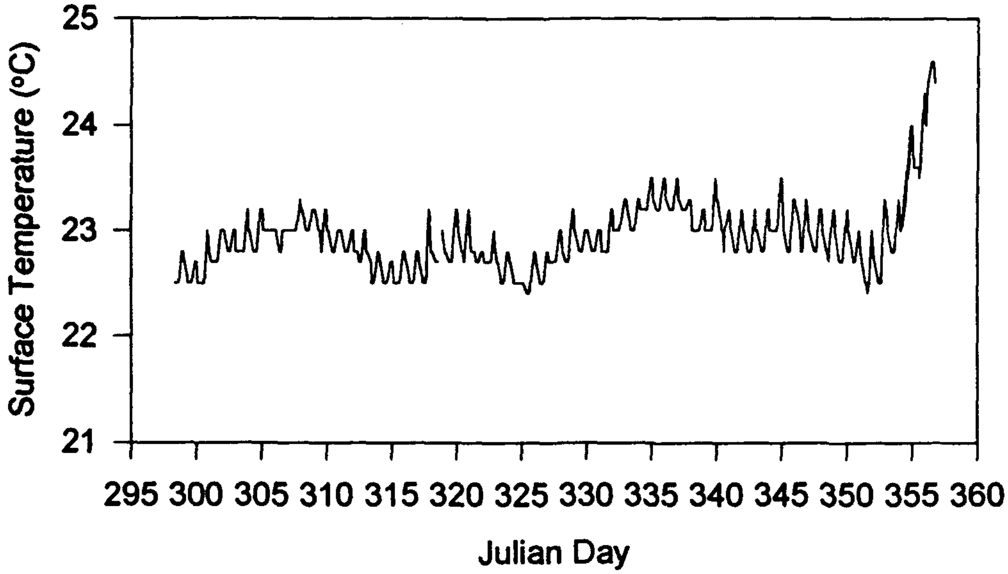


Figure 5. Sea-surface temperature variations, Buoy 4742.



5.2 Surface Irradiance. Surface irradiance at 490 nm is shown in the Figures 6-9.

Figure 6. Surface irradiance (490 nm) variations, Buoy 4745.

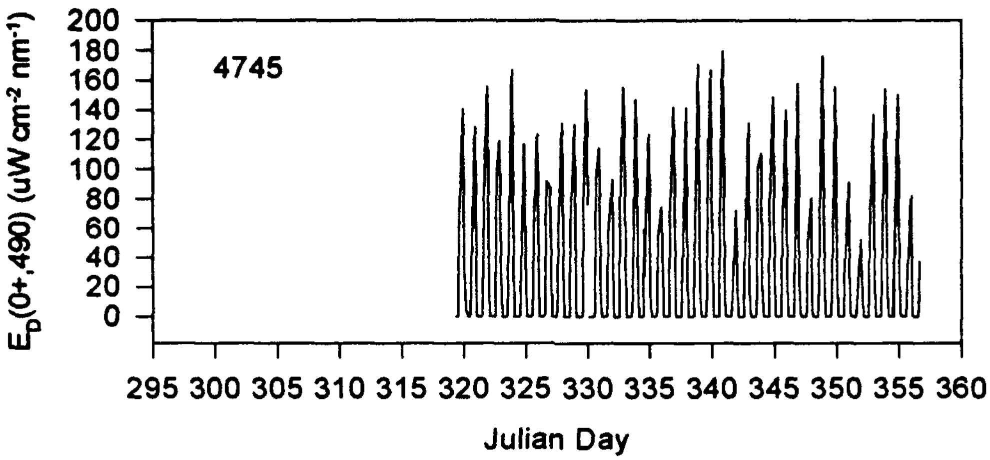


Figure 7. Surface irradiance (490 nm) variations, Buoy 4744.

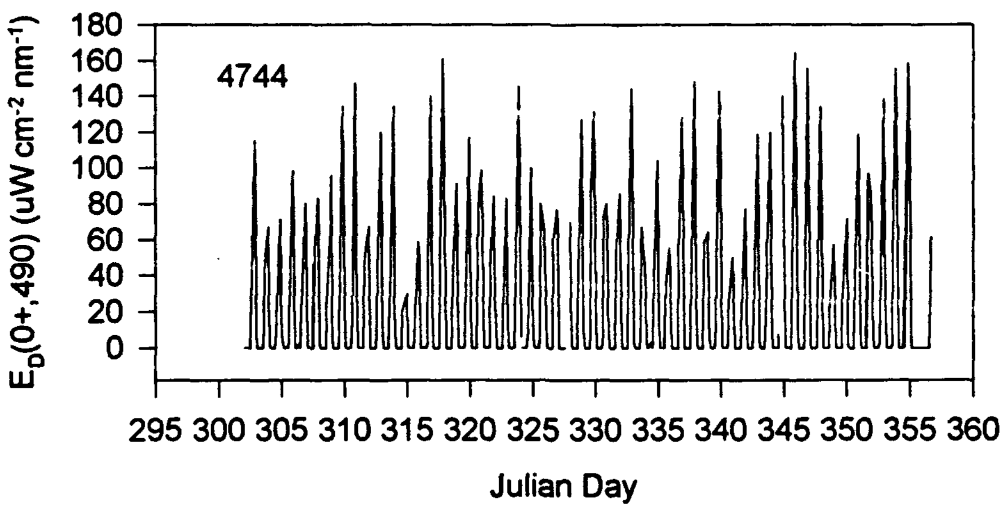


Figure 8. Surface irradiance (490 nm) variations, Buoy 4743.

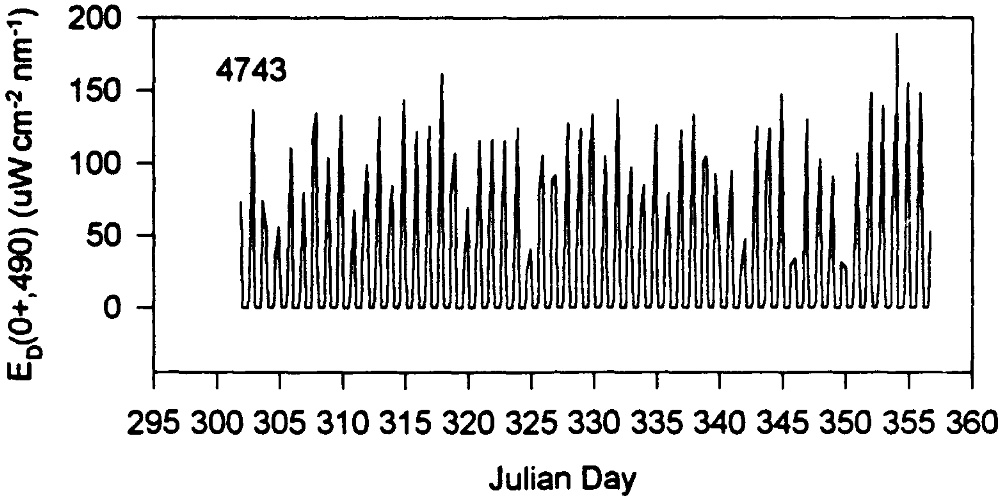
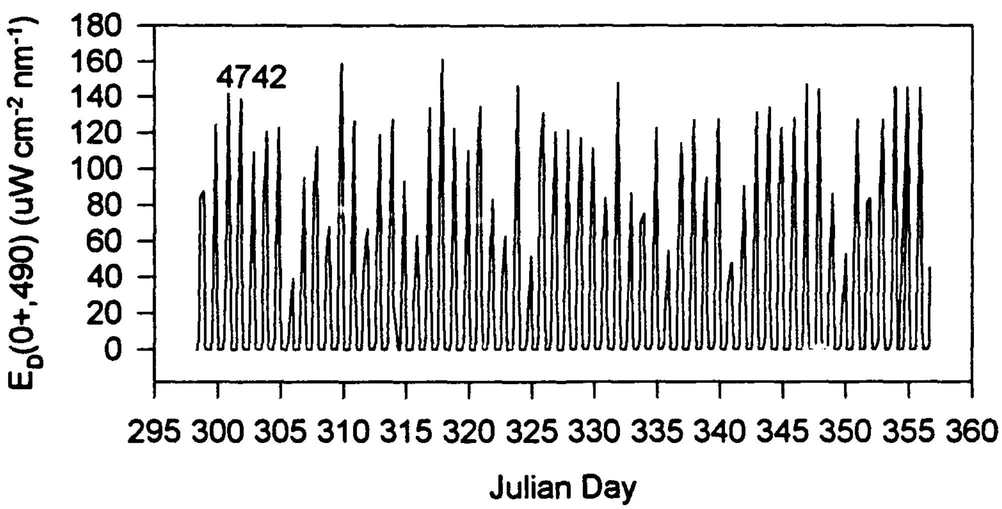


Figure 9. Surface irradiance (490 nm) variations, Buoy 4742



5.3. Surface Pigment. **Surface pigm:-nt, computed with the nominal CZCS algorithms is given in Figures 1^ 13 below.**

Figure 10. Surface pigment variations, Buoy 4745

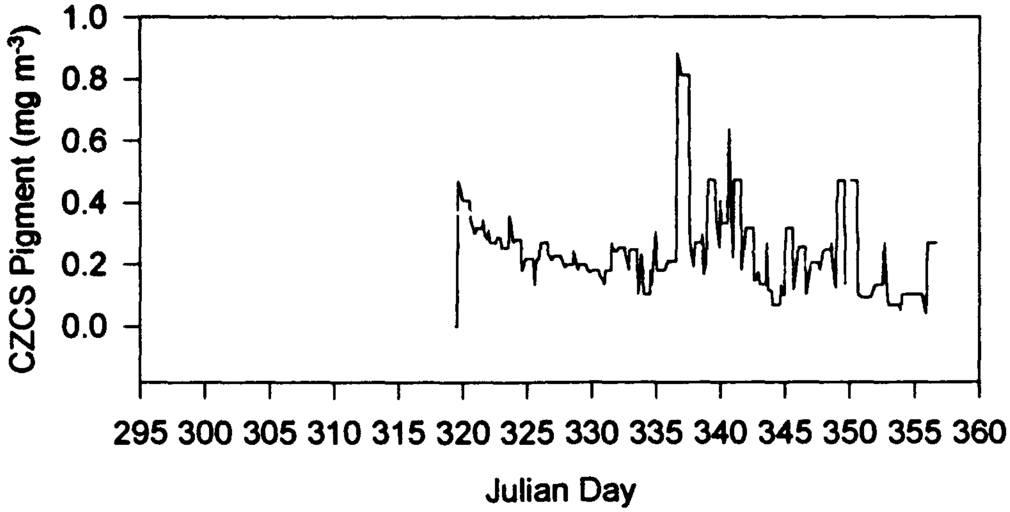


Figure 11. Surface pigment variations, Buoy 4744

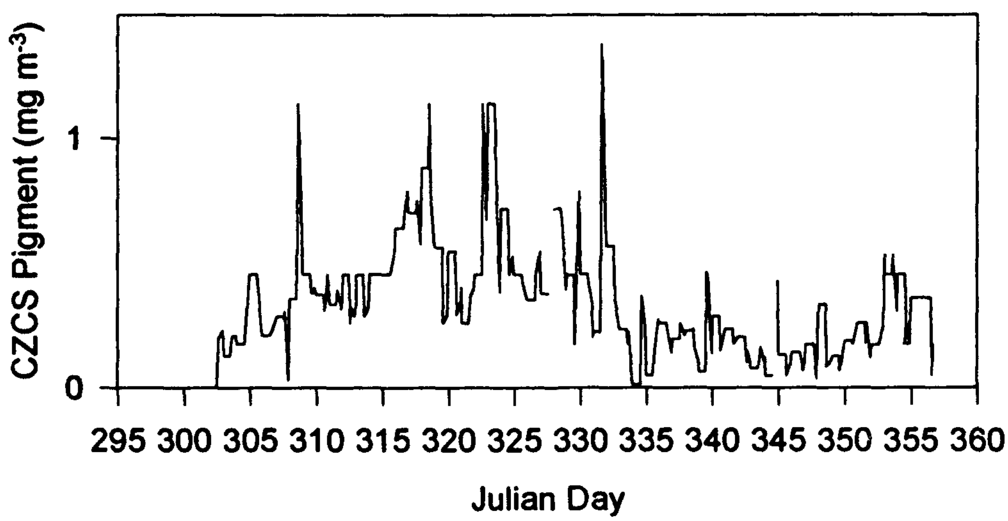


Figure 12. Surface pigment variations, Buoy 4743

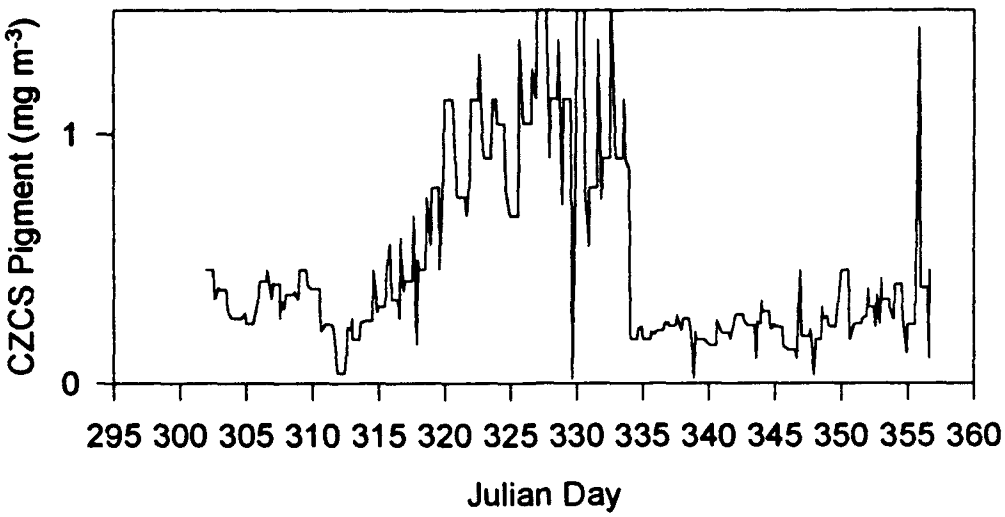
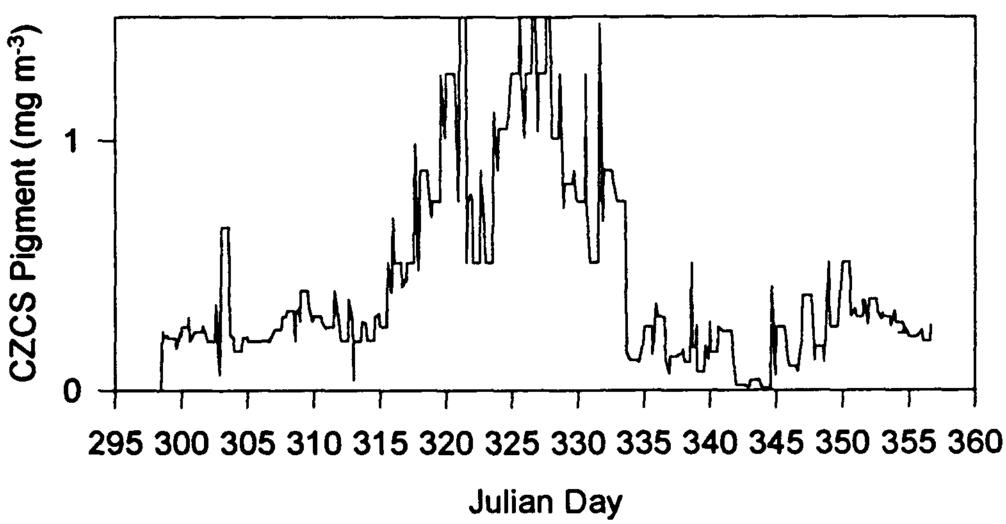


Figure 13. Surface pigment variations, Buoy 4742



5.4. Evaluation.. The buoy performance was nominal for the duration to date. SST's and pigment estimates are comparable to those measured during the IRONEX ship observations and are consistent with local climatology. Of particular note are the pigment time-series for buoys 4742-4745 which show a high degree of correspondence in time. This is the first time buoys have been deployed which can be considered essentially replicaties and these results are encouraging.

6.0 Data Availability.

Level 2 data is available for anonymous ftp on /pub/ironex on predator.ocean.dal.ca. Those wishing raw or calibrated radiance should contact marlon@predator.ocean.dal.ca and it can be made available as well.

Ocean Color Monitor (OCM)
ID 4742
User's Manual
IRONEX-Ship

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

OCM Operating Manual

Contents

Telephone: (902) 492-4780 *Facsimile: (902) 492-4781*

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

OCM Operating Manual

INTRODUCTION

The OCM (Ocean Color Monitor) is designed to be a self-contained precision optical instrument suitable for expendable launch in harsh ocean regions to remotely monitor the apparent optical properties of the upper ocean. The system can be launched from multiple platforms, including aircraft and ships of opportunity, and relays data back to the user via the ARGOS satellite system.

The OCM has a seven-channel upwelling radiance sensor and a single channel downwelling irradiance sensor. The system uses a proprietary filter/photodiode system to provide improved signal performance, ruggedness and sensor stability necessary for oceanographic use. The OCM, designed as a multiplatform module, has been used successfully as a ship-launched drifting buoy, an air-launched drifting buoy, moored systems, and tethered systems.

The rugged design of the OCM buoy resulted from its development as an aircraft gravity tube launch system which was required to withstand impacts on the ocean surface at speeds of over 50m/sec. This was verified through test launches from NASA's P3B aircraft. The OCM has also been field tested in all deployment modes in temperatures ranging from -20°C to +40°C with no degradation in performance.

The filters used in the OCM were custom designed to match the full 20nm bandwidth of the upcoming SeaWiFS satellite with an additional 10nm channel at 683nm for the measurement of solar-stimulated fluorescence. The use of custom filters allows us to precision match instruments for our customers. The spectral scans for these filters are available upon request.

A significant amount of design and field testing has gone into the OCM since the first prototype in 1989 and we are confident you will be pleased with its performance in any ocean environment.

Scott McLean,
Project Engineer

Telephone: (902) 492-4780 *Facsimile: (902) 492-4781*

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

1.0 General Specifications

Upwelling Radiance Sensor Characteristics

Sensor Model : OCR-100

Spatial Characteristics:

- Field of view: 5° (0.025 steradians) in water
  7° (0.100 steradians) in air
- Entrance aperture: 4.78 mm diameter
- Detectors: custom 13 mm^2 silicon photodiodes

Spectral Characteristics:

- Bandwidth range: 400-700 nm
- Number of channels: 7
- Spectral bandwidth: 6 channels 20 nm
  1 channel 10 nm
- Filter Type: custom low fluorescence interference
- Discrete wavelengths (centers): 412, 443, 490, 510, 555, 670, 683 nm

Optical Characteristics:

- Out of band rejection: 10^{-6}
- Out of field rejection: $5x10^{-4}$

Temporal Characteristics:

- System time constant: 0.015 seconds
- -3dB frequency: 10 Hz

Telephone: (902) 492-4780 *Facsimile: (902) 492-4781*

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

Downwelling Irradiance Sensor Characteristics

Sensor Model : ED-100

Spatial Characteristics:

- Field of view: cosine response
- Collector area: 86.0 mm^2
- Detectors: custom 13 mm^2 silicon photodiodes

Spectral Characteristics:

- Bandwidth range: 400-700 nm
- Number of channels: 1
- Spectral bandwidth: 20 nm
- Filter Type: custom low fluorescence interference
- Discrete wavelengths (center): 490nm

Optical Characteristics:

- Out of band rejection: 10^{-6}
- Cosine response: within 3% 0-60°
  within 10% 60-89°

Temporal Characteristics:

- System time constant: 0.015 seconds
- -3dB frequency: 10 Hz

Operational Characteristics: (Buoy System)

- size: 91cm X 12.5cm diameter (packaged air-launch only)
- weight: 12kg
- power: internal - 50 alkaline 'C' cells
- operating lifetime: up to 6 months
- temperature rating: -20°C to +40°C (operational)
- satellite transmitter: MetOcean MAT8602 ARGOS PTT

Telephone: (902) 492-4780 *Facsimile: (902) 492-4781*

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

2.0 Absolute Radiometric Calibration (Radiance)

Absolute radiometic radiance calibration is done using a calibrated 1000W FEL lamp on a 5m optical bar using the 'plaque method'. The lamp is powered by an Optronics 83DS current source. The flux from the lamp is normally incident on a 50 cm diffuse reflectance target standard at a distance of 200.0 cm. The instrument views the target at an angle of 45.0 deg such that the field of view of all the sensors is completely covered by the target. The calibration radiances are determined using equation 1 below:

(1) $$L(\lambda) = (E(\lambda, 50\text{cm}) / \pi) * (50.0\text{ cm} / 200.0\text{ cm})^2 * \rho(\lambda)$$

where:
- $L(\lambda)$ is the calibration radiance
- $E(\lambda, 50\text{cm})$ is the lamp calibration at 50cm
- $(50.0\text{ cm}/200.0\text{ cm})^2$ is the $1/R^2$ distance
- $\rho(\lambda)$ is the reflectance target calibration

Reflection Target: Labsphere SRT-99-180 S/N 001873
Standard Lamp: Hoffman S/N 91615
Voltmeter: HP34401A S/N 3146A09840

The voltage output of the instrument is measured on an HP34401A 6.5 digit multimeter at a sample rate of 20Hz. A 5 second average is used for the calibration output.

wavelength (nm)	lamp irradiance ($\mu W/cm^2/nm$)	target reflectance	calibration radiance ($\mu W/cm^2/nm/sr$)	OCM input (mV)	OCM dark (μV)
410	2.766	0.981	0.0540	86.45	+610
444	4.388	0.982	0.0857	115.12	+280
489	7.219	0.981	0.1409	283.10	-10
511	8.671	0.982	0.1694	298.80	-60
553	11.55	0.985	0.2263	476.70	+740
668	18.60	0.982	0.3634	1372.40	+660
684	19.36	0.983	0.3786	1274.80	-640

Table 1 - OCM ID 4742 Absolute Radiometric Calibration (Radiance)

Telephone: (902) 492-4780 *Facsimile: (902) 492-4781*

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

2.1 Absolute Radiometric Calibration (Irradiance)

Absolute radiometic irradiance calibration is done using a calibrated 1000W FEL or DXW lamp on a 5m optical bar using direct radiation from the lamp. The lamp is powered by an Optronics 83DS current source. The flux from the lamp is normally incident on the irradiance sensor cosine collector at a distance of 100.0 cm. The calibration irradiances are determined using equation 2 below:

(2) $$E(\lambda, 75cm) = E(\lambda, 50cm) * (50.0 \text{ cm} / 75.0 \text{ cm})^2$$

where:
$E(\lambda, 75cm)$ is the calibration irradiance
$E(\lambda, 50cm)$ is the lamp calibration at 50cm
$(50.0 \text{ cm}/75.0 \text{ cm})^2$ is the $1/R^2$ distance

Standard Lamp: Hoffman S/N 91604
Voltmeter: HP34401A S/N 3146A09840

The voltage output of the instrument is measured on an HP34401A 6.5 digit multimeter at a sample rate of 20Hz. A 5 second average is used for the calibration output.

wavelength (nm)	lamp irradiance ($\mu W/cm^2/nm$)	calibration irradiance ($\mu W/cm^2/nm$)	OCM input (mV)	OCM dark (μV)
489	7.219	3.208	50.96	+10

Table 2 - OCM ID 4742 Absolute Radiometric Calibration (Irradiance)

Telephone: (902) 492-4780 *Facsimile: (902) 492-4781*

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

2.2 Immersion Effect (Radiance)

Due to the difference in indices of refraction between air (where the instrument is calibrated) and water (where it is operated) a correction factor must be applied to obtain the effective in water radiances. This correction factor is referred to as the immersion factor. There are two effects contributing. First, the reduction in solid angle viewed by the sensors effectively reduces the amount of flux into the sensor. This correction is given by F1.

(2) $F1(\lambda) = (\eta_w(\lambda))^2$ where η_w is the index of refraction of water

To correct for calibration values in air, the in-water values are multiplied by the effective loss of viewing area in water (F1).

The second effect is due to the change in index of refraction at the glass/air (glass/water) interface. This correction is given by F2.

(3) $F2(\lambda) = (\eta_w(\lambda) + \eta_g(\lambda))^2 / ((\eta_w(\lambda) * (1 + \eta_g(\lambda))^2)$

where η_g is the index of refraction of window

Since the indices of refraction of water and glass are better matched, there are less reflection losses at the window. The immersion factor thus reduces the in-water values to correct for this effect.

The total immersion effect is then:

(4) $Imm(\lambda) = F2(\lambda) * F2(\lambda)$

Thus the correction for actual in-water radiance values is:

(5) $L_{water\ actual}(\lambda) = L_{water\ measured}(\lambda) * Imm(\lambda)$

Telephone: (902) 492-4780 *Facsimile: (902) 492-4781*

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

The results for the OCM are in table 3 below:

wavelength (nm)	index water	index window	immersion correction
410	1.349	1.534	1.747
444	1.347	1.534	1.741
489	1.344	1.529	1.734
511	1.343	1.529	1.732
553	1.341	1.525	1.728
668	1.338	1.520	1.720
684	1.337	1.520	1.719

Table 3 - OCM Immersion Corrections

References:

Austin, R.W. (1976), Air-water radiance calibration factor, Tech. Memo. ML-76-004t, Vis. Lab., Scripps Institute of Oceanography, 8pp.

Austin, R.W., and G. Halikas (1976), The index of refraction of seawater, SIO Ref. 76-1, Vis. Lab., Scripps Institute of Oceanography, 64pp.

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

3.0 Calibration Summary OCM ID 4742

The OCM buoy transmits the data from the sensors in hexadecimal counts from 00 to FF (0 to 255). The minimum voltage that can be read is 15mV (one count), the maximum voltage that can be read is 3.825V (255 counts). The calibration summary will give the results in counts such that the calibration factor (F) can be applied as in equations (1) and (2) below:

(1) radiance = counts * F * immersion ($\mu W/cm^2/nm/sr$)

where immersion comes from Table 3

(2) irradiance = counts * F ($\mu W/cm^2/nm$)

The calibration factor (F) is computed using equation (3)

(3) F (units/count) = calibration (units) / calibration volts (mV) * 15 (mV/count)

Sensor	Filter#	Lot#	CWL (nm)	Calibration (mV)	Cal Radiance ($\mu W/cm^2/nm/sr$)	F	Saturation* ($\mu W/cm^2/nm/sr$)
Lu412	25	1493	410	85.84	0.0540	9.436E-3	2.406
Lu443	24	1293	444	114.84	0.0857	11.19E-3	2.854
Lu490	17	1093	489	283.11	0.1409	7.465E-3	1.904
Lu510	19	1393	511	298.86	0.1694	8.502E-3	2.168
Lu555	29	1393	553	475.96	0.2263	7.132E-3	1.819
Lu670	16	1393	668	1371.7	0.3634	3.974E-3	1.013
Lu683	23	1393	684	1275.4	0.3786	4.453E-3	1.135

*saturation is calculated at 255 counts

Table 4 OCR-100 radiance sensor S/N 055

Sensor	Filter#	Lot#	CWL (nm)	Calibration (mV)	Cal Irradiance ($\mu W/cm^2/nm$)	F	Saturation* ($\mu W/cm^2/nm$)
Ed490	18	1093	489	50.95	3.208	0.944	240.8

*saturation is calculated at 255 counts

Table 5 ED-100 irradiance sensor S/N 055

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

This is a copy of the calibration file **04742A.CAL** for use with the OCM_PRO3 and OCM_PRO4 processing programs. It contains the F values used in equation (3) above to calculate physical units from the buoy digital counts.

```
# Calibration File - OCM Ship-Launch ID4742
# OCR-100 S/N 055   ED-100 S/N 055
# Calibration Date: 19 September 1993
# order: Lu683, Lu670, Lu555, Lu510, Lu490, Lu443, Lu412, Ed490
#
4.453E-3  0
3.974E-3  0
7.132E-3  0
8.502E-3  0
7.465E-3  0
11.19E-3  0
9.436E-3  0
0.944  0
```

Telephone: (902) 492-4780 *Facsimile: (902) 492-4781*

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

4.0 Deployment

The OCM has three standard launch modes: air-launch deployment, ship-launch airbag deployment, and solid collar deployment. Air-launch and ship-launch airbag deployments are by far the simplest as the package is designed to deploy automatically. These packages have salt water activated squibs which automatically deploy the buoy when in the water. The airbag floatation collar on these packages, however, is not as robust as the solid collar. The solid collar units are designed for experiments in extremely harsh conditions (such as ice) or in moored configurations where the float may chaff on mooring lines.

OCM ARGOS ID #4742 is a Ship-Launch Solid Collar Buoy. See Section 4.3 for deployment details.

4.1 Air-Launch Package Deployment

The OCM has a US Navy NAVOCEANO air-launch certification as a CMOD type buoy and is suitable for gravity tube air launch as an A-size sonobuoy package. It has been sucessfully air-launched from NASA's P3B aircraft with a 100% sucess rate. Although the package is certified over a large window of deployment speeds and altitudes, a speed of 200 knots (or less) and an altitude of at least 2000 feet is recommended to reduce the probability of package damage on impact with the water. This altitude assures that the horizontal velocity of the buoy is near zero at impact and is in a near vertical position.

The standard air-launch package is shipped in an SLC (sonobuoy launch container - 13cm diameter by 100cm long) which looks much like an AXBT package (but heavier). The ARGOS ID for the OCM inside the conatiner is written on the lid of the SLC as well as on the side. Do not store the OCM outside of the SLC before launch, as damage to the parachute may occur.

To launch the buoy, remove the retaining clip holding the lid on the SLC and pry off the SLC lid. Slide the buoy out. The ARGOS ID is also written on top of the buoy's wind flap which opens its parachute on launch. When the pilot indicates it is safe to launch, insert the buoy into the launch tube **wind flap end first (failure to do this may damage the aircraft!)** and drop the buoy out of the launch tube.

Telephone: (902) 492-4780 *Facsimile: (902) 492-4781*

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

4.2 Ship-Launch Airbag Package Deployment

The ship-launch airbag OCM is designed to be an easy self-deploying package that can be deployed from any vessel moving at full speed. It is very similar to the air-launch package except that it does not have a parachute and windflap.

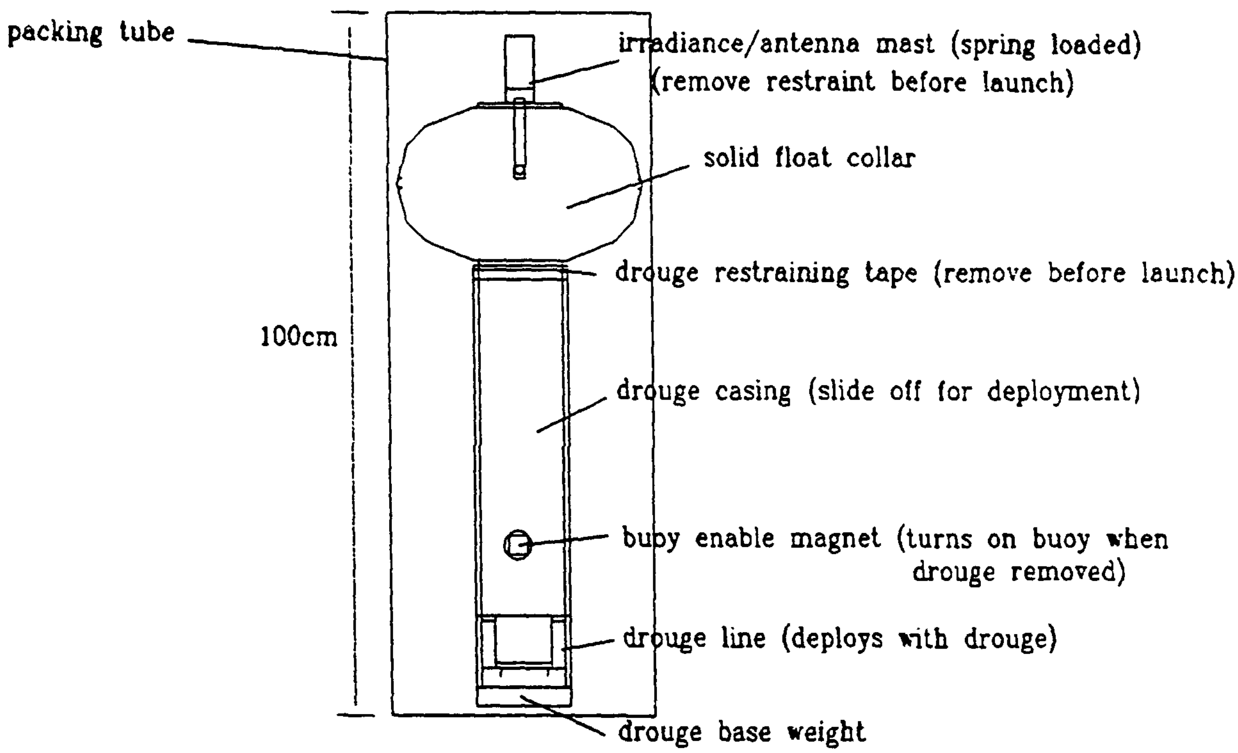
The standard ship-launch package is shipped in an SLC (sonobuoy launch container - 13cm diameter by 100cm long). The ARGOS ID for the OCM inside the container is written on the lid of the SLC as well as on the side. The side of the SLC will also be clearly marked 'ship-launch' indicating that the unit does not have a parachute.

To launch the buoy, remove the retaining clip holding the lid on the SLC and pry off the SLC lid. Slide the buoy out. **The buoy cannot be deployed inside the SLC.** The ARGOS ID is also written on top of the clam shell retainer. The buoy can be tossed over the side of the vessel for deployment. **Do not get the buoy wet on deck as it may activate the squib, blowing the clam shells off and injuring personnel.** Note that the buoy will descend to a depth of 10m before the squib fires, which takes about 10-15 seconds. The squib will fire, inflating the airbag, blowing off the clam shells, releasing the drouge and deploying the antenna. The buoy will then rise to the surface and begin transmitting. **Do not launch the buoy from a stationary vessel as the buoy may strike the vessel on its way back to the surface, damaging the antenna and/or the cosine collector.**

4.3 Ship-Launch Solid Collar Deployment

This OCM has a high density foam solid collar which can withstand very harsh conditions without being damaged. Since the collar is a fixed size, the package is not as easily deployed as the airbag type, but is more suitable for moored type applications.

The buoy comes packaged in a 37cm diameter tube, 100cm long. The package does not automatically deploy and must be deployed manually. The top cover of the tube has the buoy ARGOS ID written on it. Remove the screws which secure both covers and slide the packing tube off the buoy. The ARGOS ID is on a stamped plate which is attached to the outer casing (drouge), just above the magnet. The magnet, when removed, turns on the ARGOS transmitter. Remove the restraint holding down the antenna, exposing the irradiance cosine collector. Pull the antenna to its fullest extent. Remove tape holding the drouge (outer casing) on and slide it off (this also turns the transmitter on as the magnet slides off). The drouge can is attached to the buoy via a 20m



THIS IS NOT AN AUTOLAUNCHING PACKAGE READ LAUNCH INSTRUCTIONS CAREFULLY

Satlantic

FILENAME: PACKED_1.SKD	TITLE: Ocean Colour Monitor (OCM) Manual Solid Collar Launch		REV: 01
DRAWN BY: Scott McLean	DATE: 20 Nov 91	DRWG NO: 91-0026-MECH	

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

long cable to improve its stability. The drouge cable is packaged onto the side of the radiance sensor with wax to prevent tangling. With the vessel stopped, lower the drouge

into the water (if the vessel is not stopped the drouge acts as a sea anchor and will pull the buoy off the deck, into the water, possibly damaging it). When the drouge is fully deployed the buoy can be lowered or dropped vertically into the water. Care must be taken to not break the antenna.

Mooring schemes can be used effectively with this buoy by tightly clamping a band around the center of the float which contains attachment points. The band should dig into the float such that it cannot slide off. The most effective mooring for estuary work is a three point mooring which, with three guard floats arranged in a triangle about 10m on a side. The buoy can then be tied off to each float, providing a very stable system. The buoys have also been moored to a single float, but this destabilizes the buoy causing amplified wave induced tilts, increasing data loss to to poor transmission (see Data Processing section for details). Many other mooring schemes are possible; contact us to discuss your application.

Telephone: (902) 492-4780 ***Facsimile: (902) 492-4781***

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

5.0 Data Processing

ARGOS data is acquired from the ARGOS Data Processing Center in Landover, USA via the TYMNET system. The raw ARGOS is downloaded in the TX format (see ARGOS Users Manual section 3.2.5) to be compatible with the data processing software. The data consists of sets of data points referred to as 'hits'. Each 'hit' is a data point sent by the buoy that was stored by a satellite-based ARGOS receiver. The buoy transmits data at 90 second intervals continuously. Reception at the satellite is determined by the buoy's position and the satellite orbit (typically 10-12 times per day). The data is then sent by the satellite to the next earth station it passes over, where it is then processed by Service ARGOS to determine the platform's position. Raw telemetry (Level 0) data is downloaded every second day by the user and saved with an .ARG extension. A sample data 'hit' is shown below:

```
07188      44.672N   63.603W   0        343/1652Z-343/1651
   (  1)      2000468    301066F   301056C   301056E
              100025D    301046C   2000463      10A0
                   45         B6
```

The first line contains the platform ID (in this case 07188), the platform's estimated position in decimal degrees (in this case our test mooring), the location accuracy (single digit from 0 to 3 - 0 means accuracy not determinable, 1 within 1000m, 2 within 300m, 3 within 150m) and then the day and time of the data collection (in Julian day and time in UTC). The second day and time is the last position update.

The number in brackets `(  1)` on the second line is a compression index and is ignored by data the processing.

The remaining lines contain the raw telemetry from the buoy coded in hexadecimal. The first seven groups of numbers are the radiance channels, the eighth is the mast irradiance sensor, the ninth is the internal thermistor and the last contains the battery voltage and the tick timer. The radiance channel number contains eight hexadecimal digits. If not all digits are shown, the leading blanks are zeros. The eight digits are in groups of two hexadecimal digits bytes). The first two digits represents the channel average. This average is computed by taking the last hour of data (40 data points) and computing the average. The average is not a running average; the previous hour's average of data is transmitted while a new one is being collected. Every hour the data transmitted is updated, and the old average is discarded. The second two digits contain the minimum

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

value recorded during the hour. The third two digits contain the maximum value recorded during the hour. The last two digits contain an encoded sum of squares such that the standard deviation can be computed.

In order of occurrence, the radiance channels are 683nm, 670nm, 555nm, 510nm, 490nm, 443nm and 412nm. The average, minimum and maximum are converted to radiances using the equation:

$$\text{Radiance} = F * (\text{counts} - \text{offset}) * \text{immersion} \quad (\mu W/cm^2/nm/sr)$$

The standard deviation is computed by first recovering the sum of squares using the equation:

$$\text{sum\_squares} = (10^{(\text{counts} / 36.4)})/10000$$

The standard deviation (in volts) is then:

$$\text{Std\_Dev\_volts} = \text{sqrt}((\text{sum\_squares} - n * \text{average}^2) / (n-1))$$

In radiance units:

$$\text{Std\_Dev\_rad} = \text{Std\_Dev\_volts} * 255 / 3.825 * F * \text{immersion}$$

he calibration factors (F) are stored in the platform's calibration file which is made up of the platform ID with a .CAL extension.

The irradiance channel is the last group of digits on the second line and contains four hexadecimal digits which represent the irradiance average and sum of squares (two digits each). The irradiance sensor on all platforms is an Ed490 sensor . The sensor values are converted to physical units using the equation:

$$\text{Irradiance} = F * (\text{counts} - \text{offset}) \quad (\mu W/cm^2/nm)$$

The first number on the last line is the buoy hull temperature. This is converted from counts to degrees Celsius using the following equation:

$$\text{Temperature} = \text{counts} * 0.16 - 5$$

Telephone: (902) 492-4780 ***Facsimile: (902) 492-4781***

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Burrington Street
Halifax, Nova Scotia
Canada B3K 5X8

The last number contains the battery voltage and the tick timer.

The tick timer, stored in the lower 6 bits, indicates how many minutes it has been since the data in the transmit buffer was last updated. This value will change every transmission. To get the actual time for the middle of the transmission average, take the transmission time, subtract the tick time minutes, and then subtract 30 minutes more to get the center of the hour period.

The upper two bits indicate the battery voltage. Each bit represents 1.5 volts above 10.5 volts, at which point the platform transmitter will fail. This value must be carefully examined each day data is collected if the buoy is to be recovered. Once the batteries fail, there will be no more position data. For quick look purposes the battery voltage can be determined by looking at the first digit of the last number, as follows:

hexadecimal	binary	voltage
Fx	1111xxxx	15.0V
Ex	1110xxxx	15.0V
Dx	1101xxxx	15.0V
Cx	1100xxxx	15.0V
Bx	1011xxxx	13.5V
Ax	1010xxxx	13.5V
9x	1001xxxx	13.5V
8x	1000xxxx	13.5V
7x	0111xxxx	12.0V
6x	0110xxxx	12.0V
5x	0101xxxx	12.0V
4x	0100xxxx	12.0V
3x,2x,1x,0x	00xxxxxx	10.5V

TX Format Battery Voltage Quick Look Table

Telephone: (902) 492-4780 ***Facsimile: (902) 492-4781***

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

5.0.1 Data Transmission Errors

Due to the 256 bit per transmission limit on ARGOS there is no checksum transmitted with the data. Erroneous data can therefore occur at random and must be manually removed. Data errors will occur more frequently if the buoy is moored or restrained such that wave action causes the antenna to tilt quickly since the mooring or restraint will not follow the motion of the buoy. Best performance occurs when the buoy is allowed to free drift as its dynamics are designed for this mode of operation.

Examples of these errors (indicated by circled data records) can be seen in Appendix 3 where OCM_PRO4 has been run on file SEP05.ARG. Buoy 04742 was moored with a single point mooring in Monterey Bay in 50m of water during a period of strong onshore winds. The platform antenna was moving rapidly during this period, resulting in poor communications. The data from all of the good transmissions shows very consistent data. Buoy 04745 was air-launched into the equatorial Pacific and was free drifting; it occasionally gets a bad transmission.

To screen bad transmissions, first check the temperature. Bad transmissions often have very high or very low temperatures. Check for other inconsistencies such as very high radiances or irradiances. Bad transmissions will be obvious when you plot your data. If you have any questions about this, call us.

Telephone: (902) 492-4780 ***Facsimile: (902) 492-4781***

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

5.1 Data Processing with OCM_PRO3.C

Two sample programs are included in the Appendix 1 OCM_PRO3.C and OCM_PRO4.C. These programs read TX formatted ARGOS data and produce calibrated ASCII output files for each platform transmitter. These programs are the ones that we use to process our own data, and are provided free of charge. Please feel free to modify them as you require. We use OCM_PRO3.C to apply calibrations to our data and then use MATLAB to produce final output and run models.

Here is a brief decription of OCM_PRO3.C. Source code is listed in Appendix 1 and examples are provided in Appendix 3:

Program OCM_PRO3.C

written by: Scott McLean
ver date: Jan 15/93
distribution: unlimited
(c) Satlantic Inc. 1993

valid for buoy data from 05 Aug 92 to date

This program ingests OCM type buoy datafiles downloaded from the ARGOS Processing Center. The data must be downloaded from the processing center in TX format using the PRV command (refer to ARGOS Users Manual section 3.2.5).

The program extracts the buoy data from these files and produces an ASCII file with calibrated data in physical units. Any number of such files can be read. The data in each file is scanned and each data transmission will be processed, all extraneous characters (for example, the login) are ignored. Only one platform can be processed at a time, if you have three platforms, you must run the program three times.

This program produces output files which are very wide (exceeding 256 characters), if this is too large for your other processing programs, use OCM_PRO4.C which does not include min, max and standard deviation values (only the averages) and produces a more compact output file.

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

Required inputs:

1) INFILES.LST

This file contains a list of all files that are to be processed. The program assumes all raw data files have the .ARG extension.

2) .ARG files

These files contain the data obtained on various days from the ARGOS processing center in TX format. All files must have the .ARG extension.

3) .CAL files

These files contain the calibration data for each platform. The first 5 characters of this file name are the platform ARGOS ID number. The program will prompt you for the calibration file name. It will remove the first five characters from the calibration file name and then search all data files in INFILES.LST for data hits containing this ID.

4) Output file

This is an ASCII output file containing the data for the platform requested. The program will prompt you for a file name. If the file already exists, it will be overwritten.

Telephone: (902) 492-4780 ***Facsimile: (902) 492-4781***

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

5.2 Software

A disk is provided with each system. The disk contains the following files:

OCM_PRO3.C - source code for processing program
OCM_PRO4.C - source code for short processing program
04252A.CAL - calibration file for processing data from OCM 4252
[EXAMPLES] - examples directory
- OCM_PRO3.EXE - executable
- OCM_PRO4.EXE - executable
- SEP05.ARG - sample multiplatform data file
- INFILES.LST - sample batch processing file
- 04742A.CAL - sample calibration file
- 04745A.CAL - sample calibration file
- 04742_05.DAT - SEP05 processed using OCM_PRO4
- 04745_05.DAT - SEP05 processed using OCM_PRO4

The file SEP05.ARG was collected during our JGOFS aircraft mission in 1992, platform 04742 was moored in Monterey Bay, California. Platforms 04744 and 04745 were air-launched over the equatorial Pacific.

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

6.0 General Information

6.1 Anti-Biofoulant:

Toxic anti-biofoulants are illegal in Canada, thus there is no anti-biofoulant on the sensor. However, we do recommend that one be used, if it is legal in your region. The recommended product for coating optics is a spray can of Classic Yacht manufactured by Philadelphia Resins which is sold in most marine stores and may be approved by the EPA for use on outboard motors in some areas. Note that the compound contains Tributytin Methacrylate and is EXTREMELY toxic. The warning labels should be read carefully and the appropriate respirator, gloves, goggles, and envirosuit should be worn. Also note that some batches of spray cans of the material are not appropriate for use on optics. You should test spray on a clear surface and let dry, if it appears hazy blue, then it is not appropriate. Satlantic is not responsible for the results when using such compounds.

6.2 Problems?

If you have any problems or questions about the instrument, call Satlantic at (902) 492-4780 between 9am and 5pm Atlantic Time, or FAX us at (902) 492-4781.

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

APPENDIX 1

Source Code Listings for:

OCM_PRO3.C

OCM_PRO4.C

Telephone: (902) 492-4780 ***Facsimile: (902) 492-4781***

```
/* Program   OCM_PRO3.C

        This program ingests datafiles captured from the
   ARGOS Processing Center. The data must be downloaded
   from the processing center in TX format using the PRV
   command (refer to ARGOS Users Manual section 3.2.5.
        The program extracts the buoy data from these files
   and produces an ASCII file with calibrated data in
   physical units.


   written by: Scott McLean

   rlease version 2.0

   ver date: Jan 15/93 for SYMPHONY formatted files
                       June 8/93 modified for ARGOS TX formatted files

   distribution: unlimited

   (c) Satlantic Inc. 1993

        Satlantic Inc.
        3295 Barrington St.
        Halifax, NS
        CANADA B3K 5X8

        telephone: (902) 492-4780
        facsimile: (902) 492-4781


   valid for buoy data from 05 Aug 92 - date


              This program produces output files which are very wide
   (exceeding 256 characters), if this is too large, use OCM_PRO4.C
   which does not include min, max and standard deviation values (only
   the averages) and produces a more compact output file.


              This program reads a list of data files which contain
   TX formated ARGOS data from OCM type buoys. Any number of such
   files can be read. The data in each file is scanned and each
   data transmission will be processed, all extraneous characters
   (for example, the login) are ignored. Only one platform can be
   processed at a time, if you have three platforms, you must run
   program three times.


   Required inputs:

        1) INFILES.LST

              This file contains a list of all files that are to be
        processed. File names must not have exetnsion in this list.
        The program assumes all raw data files have the .ARG extension.

        2) .ARG files

               These files contain the data obtained on various days from
```

```
        the ARGOS processing center in TX format. All files must have
        the .ARG extension.


     3) .CAL files

             These files contain the calibration data for each platform.
        The first 5 characters of this file name are the platform ARGOS
        ID number. The program will prompt you for the calibration file
        name. It will remove the first five characters from the
        calibration file name and then search all data files in INFILES.LST
        for data hits containing this ID.

     4)  output file

             This is an ASCII output file containing the data for the platfo
        requested. The program will prompt you for a file name. If the file
        already exists, it will be overwritten.

                                                            */


#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <string.h>
#include <stdlib.h>

#define num_chans 7
#define COMMENT '#'
#define IRRAD 7



/*  FUNCTION PROTOTYPES */

double strtohex(char hexchar);
double hextod(char string[3]);
double cal_rad(double counts, int channel);
double cal_irrad(double counts);
double cal_temp(double counts);
double exp10(double arg);
double stand_dev(double sum_squares, double mean);
double get_tick(double tick);
double get_volts(double tick);
int get_calibration_vals();
int hit(char line[80], char platform[6]);


/*  GLOBAL VARIABLES */

 double scale[8],offset[8];   /* calibrations */

 FILE *CAL_data;
 char platform[6];


/* FUNCTIONS */
```

```
/* FUNCTION  STRTOHEX converts a character '0'..'F' into decimal 0..15 */

double strtohex(char hexchar)
{
 double hex;

 switch ( hexchar )
  {
        case ' ': hex = 0; break;
        case '0': hex = 0; break;
        case '1': hex = 1; break;
        case '2': hex = 2; break;
        case '3': hex = 3; break;
        case '4': hex = 4; break;
        case '5': hex = 5; break;
        case '6': hex = 6; break;
        case '7': hex = 7; break;
        case '8': hex = 8; break;
        case '9': hex = 9; break;
        case 'A': hex = 10; break;
        case 'B': hex = 11; break;
        case 'C': hex = 12; break;
        case 'D': hex = 13; break;
        case 'E': hex = 14; break;
        case 'F': hex = 15; break;
   }
 return hex;

/* END FUNCTION STRTOHEX  */

}


/* FUNCTION HEXTOD converts a string '00'..'FF' to decimal 0..255 */

double hextod(char string[3])

{
 double hex;

 hex = strtohex(string[0]) * 16 + strtohex(string[1]);
 return hex;

/* END FUNCTION HEXTOD */

}

/* FUNCTION GET_TICK computes time since last transmit buffer update */

double get_tick(double tick)
{
```

```
/* get time since last transmit buffer update in mimutes */

double mins;
unsigned char test;

test = tick;
test = test & 0x3F; /* mask off battery voltage in upper two bits */
mins = test;
mins = mins * 1; /* each tick is 1 minute */
return mins;

/*  END FUNCTION GET_TICK */

}

/* FUNCTION GET_VOLTS calculates current battery voltage */

double get_volts(double tick)
{

 /* battery voltage in upper two bits, each bit represents 1.5V above
       10.5V  */

 double volts;
 unsigned char test;

 test = tick;
 test = test >> 6;
 volts = test;
 volts = volts * 1.5 + 10.5;

 return volts;

/* END FUNCTION GET_VOLTS */

}

/* FUNCTION CAL_RAD calculates upwelling radiances from the calibration
                                       values, include the immersion effects */

double cal_rad(double counts, int channel)

{
 double rad;
 double immersion[] = {1.719, 1.720, 1.728, 1.732, 1.734, 1.741, 1.747};

 channel = channel - 1;     /*  channel 0 = Lu683  */

 rad = scale[channel] * (counts - offset[channel]) * immersion[channel];
```

```
return rad;

/* END FUNCTION CAL_RAD */


/* FUNCTION CAL_IRRAD calculates downwelling irradiance from mast sensor
                                             calibration values     */

double cal_irrad(double counts)

{
 double irrad;

 irrad = scale[IRRAD] * (counts - offset[IRRAD]);  /* in uW/cm^2/nm */

 return irrad;

/* END FUNCTION CAL_IRRAD */

}


/* FUNCTION CAL_TEMP  calculates the buoy hull temperature */

double cal_temp(double counts)

{
 double temp;
 temp = counts * 0.16 - 5.0;  /* degrees C */

 return temp;

/* END FUNCTION CAL_TEMP */

}


/* FUNCTION EXP10 calculates 10^X   */

double exp10(double arg)

{ double val;

  val = exp ( arg * log(10) );

  return val;

 /* END FUNCTION EXP10  */

}
```

```
/* FUNCTION STAND_DEV  computes the encoded standard deviation */


double stand_dev(double sum_squares, double mean)
/* mean should be in volts */
{
 double dev, var;
 double n = 40.0;
 double scale_fact = 36.4;

 mean= 3.825 / 255 * mean;
 sum_squares = sum_squares / scale_fact;
 sum_squares = exp10(sum_squares) / 10000;   /* sum of squares */
 var = ( sum_squares  - (mean * mean) * n ) / (n-1);  /* variance */
 if (var<0)
   dev=0;
 else
   dev = sqrt(var);  /* standard deviation in volts */

 dev = dev * 255.0 / 3.825; /* convert back to counts for cal_rad */
 return dev;


/* END FUNCTION STAND_DEV  */

}


/* PROCEDURE PROCESS_DATA_1 processes line 1 of a data hit in TX format */


void process_data_1(char line[80],
                          double *lat,
                          double *lon,
                          double *class,
                          double *day,
                          double *pass_time)

 {
  char  lat_str[22]="";
  char  lon_str[8]="";
  char  class_str[8]="";
  char  day_str[8]="";
  char  pass_time_str[8]="";
  char  lat_hem;
  char  lon_hem;
  char *end_str;


  /* extract parameters from input string */

  sscanf(line,"%*8c%6c%c%*c%7c%c",lat_str,&lat_hem,lon_str,&lon_hem);
```

```
sscanf(line,"%*27c%7s",class_str);
sscanf(line,"%*40c%3s",day_str);
sscanf(line,"%*44c%4s",pass_time_str); /**/

/* convert strings to numerics */

 *lat = strtod(lat_str, &end_str);
 *lon = strtod(lon_str, &end_str);
 *class = strtod(class_str, &end_str);
 *day = strtod(day_str, &end_str);
 *pass_time = strtod(pass_time_str, &end_str);

 *lat *= 2*(lat_hem == 'N')-1;
 *lon *= 2*(lon_hem == 'E')-1;

/* END procedure  PROCESS_DATA_1 */
}


/* PROCEDURE PROCESS_DATA_2 processes line 2 of a data hit in TX format */

void process_data_2(char line[80],
                            double mins[],
                            double maxs[],
                            double aves[],
                            double stddevs[]) /**/

{ char ch1_str[] = "12345678";
  char ch2_str[] = "12345678";
  char ch3_str[] = "12345678";
  char ch4_str[] = "12345678";
  char min_str[] = "  ", max_str[] = "  ",
       ave_str[] = "  ", stddev_str[] = "  ";
  char *end_str;


  sscanf(line,"%*9c%8c%*5c%8c%*5c%8c%*5c%8c",
         ch1_str, ch2_str, ch3_str, ch4_str);

  /* process channel 1   Lu683   */

  sscanf(ch1_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

  mins[1] = hextod(min_str);
  maxs[1] = hextod(max_str);
  aves[1] = hextod(ave_str);
  stddevs[1] = hextod(stddev_str);


  /* process channel 2   Lu670   */

  sscanf(ch2_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);
```

```
mins[2] = hextod(min_str);
maxs[2] = hextod(max_str);
aves[2] = hextod(ave_str);
stddevs[2] = hextod(stddev_str);

/* process channel 3   Lu555   */

sscanf(ch3_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

mins[3] = hextod(min_str);
maxs[3] = hextod(max_str);
aves[3] = hextod(ave_str);
stddevs[3] = hextod(stddev_str);

/* process channel 4   Lu510   */

sscanf(ch4_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

mins[4] = hextod(min_str);
maxs[4] = hextod(max_str);
aves[4] = hextod(ave_str);
stddevs[4] = hextod(stddev_str);


/* END procedure PROCESS_DATA_2 */

}


/* PROCEDURE PROCESS_DATA_3 processes line 3 of a data hit in TX format */


void process_data_3(char line[80],
                    double mins[],
                    double maxs[],
                    double aves[],
                    double stddevs[],
                    double *irrad_ave,
                    double *irrad_std) /**/

{ char ch1_str[] = "12345678";
  char ch2_str[] = "12345678";
  char ch3_str[] = "12345678";
  char ch4_str[] = "1234";
  char min_str[] = "  ", max_str[] = "  ",
       ave_str[] = "  ", stddev_str[] = "  ";
  char *end_str;


  sscanf(line,"%*9c%8c%*5c%8c%*5c%8c%*9c%4c",
         ch1_str, ch2_str, ch3_str, ch4_str);

  /* process channel 5   Lu490 */

  sscanf(ch1_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

  mins[5] = hextod(min_str);
```

```
 maxs[5] = hextod(max_str);
 aves[5] = hextod(ave_str);
 stddevs[5] = hextod(stddev_str);

 /* process channel 6   Lu443 */

 sscanf(ch2_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

 mins[6] = hextod(min_str);
 maxs[6] = hextod(max_str);
 aves[6] = hextod(ave_str);
 stddevs[6] = hextod(stddev_str);

 /* process channel 7   Lu412 */

 sscanf(ch3_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

 mins[7] = hextod(min_str);
 maxs[7] = hextod(max_str);
 aves[7] = hextod(ave_str);
 stddevs[7] = hextod(stddev_str);

 /* process irradiance  Ed490  */

 sscanf(ch4_str,"%2c%2c",ave_str,stddev_str);

 *irrad_ave = hextod(ave_str);
 *irrad_std = hextod(stddev_str);


/* END procedure PROCESS_DATA_3 */

}


/* PROCEDURE PROCESS_DATA_4 processes line 4 of a data hit in TX format */


void process_data_4(char line[80],
                         double *temp,
                         double *tick) /**/


{ char ch1_str[] = "12";
  char ch2_str[] = "12";
  char *end_str;


  sscanf(line,"%*15c%2c%*11c%2c",ch1_str, ch2_str);


  /* process temperature */

  *temp = hextod(ch1_str);
  *tick = hextod(ch2_str);


 /* END procedure PROCESS_DATA_4 */
```

```
/* FUNCTION  LOCATION_CLASS  reports accuracy of the platform location
                                                    in meters from the ARGO

double location_class(double class)

{
 double accuracy;

 accuracy =0.0;
 if (class == 1.0)
        accuracy = 1000.0;
 if (class == 2.0)
        accuracy = 350.0;
 if (class == 3.0)
        accuracy = 150.0;
 return accuracy;

/* END FUNCTION LOCATION_CLASS */


/* FUNCTION DEC_DATE  computes the decimal day from the day, hour, minute */

double dec_date(double day, double time)

{

 double hour, minutes, dec_day;

 hour = floor(time/100.0);
 minutes = time - hour * 100;
 dec_day = day + hour/24.0 + minutes/24.0/60.0;
 return dec_day;

/*  END FUNCTION DEC_DATE */

}


/* PROCEDURE PROCESS_HIT uses data extracted by the PROCESS_DATA_ procedures,
                                                   applies calibration data and wr
                                                   to the output file    */


void process_hit(FILE *PROCESS_data,
                 double day,
                 double time,
                 double lat,
                 double lon,
                 double class,
                 double aves[],
```

```
                double mins[],
                double maxs[],
                double stddevs[],
                double irrad_ave,
                double irrad_stddev,
                double temp,
                double tick)

/* process counts and produce physical units for each parameter */


double accuracy;
int channel,i;
double std_dev,battery;
double ave, min, max,tmp, minutes;
double date;

accuracy= location_class(class);
date=dec_date(day,time);


fprintf(PROCESS_data,"%s %9.4lf %6.3lf %6.3lf %5.0lf",platform,
                date,lat,lon,accuracy);

 /* apply calibrations to radiance channels */

for (channel=1; channel < 8; channel++)
{
  i=channel;

  tmp=aves[i];
  ave = cal_rad(tmp, channel);

  tmp=mins[i]; min = cal_rad(tmp, channel);

  tmp=maxs[i]; max = cal_rad(tmp, channel);

  tmp=stddevs[i];
  std_dev = stand_dev(tmp, aves[i]);
  std_dev = cal_rad(std_dev, channel);


  fprintf(PROCESS_data," %6.4lf %6.4lf %6.4lf %6.4lf", ave, min, max,
                 std_dev);

}

/* apply calibration to irradiance data */

ave = cal_irrad(irrad_ave);
std_dev = stand_dev(irrad_stddev,irrad_ave);
std_dev= cal_irrad(std_dev);

temp = cal_temp(temp);
minutes = get_tick(tick);
battery = get_volts(tick);

fprintf(PROCESS_data," %5.1lf %5.1lf %4.1lf %4.1lf %4.1lf\n",ave,std_dev,
```

```
                temp,battery,minutes);

/* END PROCEDURE PROCESS_HIT  */
}


/*  PROCEDURE GET_CALIBRATION_VALS  reads in calibration data */

int get_calibration_vals()

{
 int error,i;
 char line[80]="";

 char first_char;


 i=0;
 error=0;

 while (!feof(CAL_data))
  {
        line[0] = 0;
        fgets(line,79,CAL_data);

        #if defined DEBUG
        printf("%s",line);
        #endif


        if (strlen(line) == 0 || line[0] == COMMENT) continue;
        sscanf(line,"%le %lf",scale+i,offset+i);

        #if defined DEBUG
        printf("%le %lf\n",scale[i],offset[i]);
        #endif

        i++;
        if ((i>8) && (line[0] != '\n'))
         {
          printf(" ERROR in calibration file - too many cal vals\n");
          error=-1;
         }
  }
  if (i<8)
   {
        printf(" ERROR in calibration file - too few cal vals\n");
        error=-1;
   }
  fclose(CAL_data);
  return error;

 /*  END PROCEDURE GET_CALIBRATION_VALS */
```

```
/* PROCEDURE HIT */

int hit(char line[80],
                char platform[6])

{

char check[9]="";

 if (strstr(line,platform) == NULL) return 0;  /* if ID not in line skip */

 sscanf(line,"%10s",check);   /* check first 10 chars for ID only */

 if (strstr(check, platform) == NULL)
   return 0;
 else
   return 1;

}

/**   PROCEDURE MAIN   **/

main()

{
 FILE *ARGOS_data;    /* current ARGOS file */
 FILE *PROCESS_data;  /* output file */
 FILE *PROCESS_LIST;  /* list of files to process */

 char file_list[] = "infiles.lst";
 char infile_ext[] = ".ARG";
 char infile_name[13];
 char outfile_name[13];
 char cal_file_name[13];
 char line[80] = "";
 int line_no = 0;
 int data_hits = 0;

 double lat,lon,class,day,time;
 double mins[8], maxs[8], aves[8], stddevs[8];
 double irrad_ave, irrad_std;
 double temp;
 double tick_time;
 int i;


 /*  read in list of files to .ARG files to process */

 if ((PROCESS_LIST = fopen(file_list,"r")) == NULL)
  {
   printf(" File INFILES.LST must exist in current directory\n");
```

```c
  printf(" and must contain a list of files to process!\n");
  return -1;
 }

/* get output file name for calibrated data */

puts(" ENTER output file name:");
scanf("%13s",outfile_name);
if ((PROCESS_data = fopen(outfile_name,"w")) == NULL)
 {
  printf(" ERROR creating %s \n",outfile_name);
  return -1;
 }

/* read in calibration file for current platform */

puts(" ENTER platform calibration file name:");
scanf("%13s",cal_file_name);

if ((CAL_data = fopen(cal_file_name,"r")) == NULL)
 {
  printf(" ERROR calibration file  %s not found\n",cal_file_name);
  return -1;
 }


if (get_calibration_vals() == -1)
 {
        printf(" Program Terminated \n");
        return -1;    /*  read cal file */
 }


/* extract platform name from first 5 characters of cal file name */

sscanf(cal_file_name,"%5s",platform);


/* put text header on output file */

fprintf(PROCESS_data,"  ID       DAY      LAT     LON   ACCUR");
fprintf(PROCESS_data," Lu683ave  min     max     std");
fprintf(PROCESS_data," Lu670ave  min     max     std");
fprintf(PROCESS_data," Lu555ave  min     max     std");
fprintf(PROCESS_data," Lu510ave  min     max     std");
fprintf(PROCESS_data," Lu490ave  min     max     std");
fprintf(PROCESS_data," Lu443ave  min     max     std");
fprintf(PROCESS_data," Lu412ave  min     max     std");
fprintf(PROCESS_data," ED490ave std Temp Vbat Tick\n");


/* scan .ARG files and extract data for current platform */

while ( !feof(PROCESS_LIST) )
 {
        line[0]=0;
```

```
        infile_name[0]=0;
        fgets(line,80,PROCESS_LIST);

        if ( (line[0] == '\0') || (line[0] == ' ') || line[0] == '\n' ) continu
        sscanf(line,"%s",infile_name);

 /* scan current .ARG file for current platform */

if ((ARGOS_data = fopen(infile_name,"r")) != NULL)
{
 fgets(line,80,ARGOS_data);    /* read line from file */
 line_no++;
 while ( !feof(ARGOS_data) )
  {

        /* check current line for correct plaform ID */

        if (hit(line,platform))  /* skip line if not data hit */
         {

          data_hits++;
          printf(" processing hit %3.0i platform %s\n",data_hits,platform);
          process_data_1(line, &lat, &lon, &class, &day, &time);

          fgets(line,80,ARGOS_data);
          process_data_2(line, mins, maxs, aves, stddevs);
          line_no++;

          fgets(line,80,ARGOS_data);
          process_data_3(line, mins, maxs, aves, stddevs, &irrad_ave, &irrad_st
          line_no++;

          fgets(line,80,ARGOS_data);
          process_data_4(line, &temp, &tick_time);
          line_no++;

          process_hit(PROCESS_data,day,time,lat,lon,class,aves,mins,maxs,
                   stddevs,irrad_ave,irrad_std,temp,tick_time);
         }
         fgets(line,80,ARGOS_data);    /* read next line */
         line_no++;
  }
  line_no--;
  fclose(ARGOS_data);
 }
  else
        printf(" Error file: %s not found\n",infile_name);

 } /* end while processing */

 fclose(PROCESS_LIST);
 fclose(PROCESS_data);
 printf(" %i lines processed\n",line_no);
 printf(" %i data hits processed\n",data_hits);
 return 0;


/***  END MAIN  ***/
```

Program OCM_PRO4.C

This program ingests datafiles captured from the ARGOS Processing Center. The data must be downloaded from the processing center in TX format using the PRV command (refer to ARGOS Users Manual section 3.2.5.

The program extracts the buoy data from these files and produces an ASCII file with calibrated data in physical units.

written by: Scott McLean

release version 2.0

ver date: Jan 15/93 for SYMPHONY formatted files
June 8/93 modified for ARGOS TX formatted files

distribution: unlimited

Satlantic Inc.
3295 Barrington St.
Halifax, NS
CANADA B3K 5X8

telephone: (902) 492-4780
facsimile: (902) 492-4781

valid for buoy data from 05 Aug 92 - date

This program produces output files which only contains the average values to produce a more compact output file. Use OCM_PRO3.C if all of the data including min, max and stanadrd deviation is desired.

This program reads a list of data files which contain TX formated ARGOS data from OCM type buoys. Any number of such files can be read. The data in each file is scanned and each data transmission will be processed, all extraneous characters (for example, the login) are ignored. Only one platform can be processed at a time, if you have three platforms, you must run program three times.

Required inputs:

1) INFILES.LST

This file contains a list of all files that are to be processed. File names must not have exetnsion in this list. The program assumes all raw data files have the .ARG extension.

2) .ARG files

These files contain the data obtained on various days from

```
/* FUNCTION  STRTOHEX converts a character '0'..'F' into decimal 0..15 */

double strtohex(char hexchar)
{
double hex;

switch ( hexchar )
 {
        case ' ': hex = 0; break;
        case '0': hex = 0; break;
        case '1': hex = 1; break;
        case '2': hex = 2; break;
        case '3': hex = 3; break;
        case '4': hex = 4; break;
        case '5': hex = 5; break;
        case '6': hex = 6; break;
        case '7': hex = 7; break;
        case '8': hex = 8; break;
        case '9': hex = 9; break;
        case 'A': hex = 10; break;
        case 'B': hex = 11; break;
        case 'C': hex = 12; break;
        case 'D': hex = 13; break;
        case 'E': hex = 14; break;
        case 'F': hex = 15; break;
  }
return hex;

/* END FUNCTION STRTOHEX  */

}


/* FUNCTION HEXTOD converts a string '00'..'FF' to decimal 0..255 */

double hextod(char string[3])
{
 double hex;

 hex = strtohex(string[0]) * 16 + strtohex(string[1]);
 return hex;

/* END FUNCTION HEXTOD */

}

/* FUNCTION GET_TICK computes time since last transmit buffer update */

double get_tick(double tick)
 {
```

```
/* get time since last transmit buffer update in mimutes */

double mins;
unsigned char test;

test = tick;
test = test & 0x3F; /* mask off battery voltage in upper two bits */
mins = test;
mins = mins * 1; /* each tick is 1 minute */
return mins;

/*  END FUNCTION GET_TICK */

}


/* FUNCTION GET_VOLTS calculates current battery voltage */


double get_volts(double tick)


 /* battery voltage in upper two bits, each bit represents 1.5V above
        10.5V  */


 double volts;
 unsigned char test;

 test = tick;
 test = test >> 6;
 volts = test;
 volts = volts * 1.5 + 10.5;

 return volts;


/* END FUNCTION GET_VOLTS */

}


/* FUNCTION CAL_RAD calculates upwelling radiances from the calibration
                                            values, include the immersion effects */


double cal_rad(double counts, int channel)

{
 double rad;
 double immersion[] = {1.719, 1.720, 1.728, 1.732, 1.734, 1.741, 1.747};


 channel = channel - 1;     /*  channel 0 = Lu683   */

 rad = scale[channel] * (counts - offset[channel]) * immersion[channel];
```

```
return rad;

/* END FUNCTION CAL_RAD */
}

/* FUNCTION CAL_IRRAD calculates downwelling irradiance from mast sensor
                                              calibration values     */

double cal_irrad(double counts)

 double irrad;
 irrad = scale[IRRAD] * (counts - offset[IRRAD]);  /* in uW/cm^2/nm */
 return irrad;

/* END FUNCTION CAL_IRRAD */
}

/* FUNCTION CAL_TEMP  calculates the buoy hull temperature */

double cal_temp(double counts)

{
 double temp;
 temp = counts * 0.16 - 5.0;  /* degrees C */
 return temp;
/* END FUNCTION CAL_TEMP */
}

/* FUNCTION EXP10 calculates 10^X  */

double exp10(double arg)
{ double val;
  val = exp ( arg * log(10) );
  return val;

/* END FUNCTION EXP10  */
}
```

```
/* FUNCTION STAND_DEV  computes the encoded standard deviation */

double stand_dev(double sum_squares, double mean)
/* mean should be in volts */
{
double dev, var;
double n = 40.0;
double scale_fact = 36.4;

mean= 3.825 / 255 * mean;
sum_squares = sum_squares / scale_fact;
sum_squares = exp10(sum_squares) / 10000;   /* sum of squares */
var = ( sum_squares  - (mean * mean) * n ) / (n-1);  /* variance */
if (var<0)
  dev=0;
else
  dev = sqrt(var);  /* standard deviation in volts */

dev = dev * 255.0 / 3.825; /* convert back to counts for cal_rad */
return dev;


/* END FUNCTION STAND_DEV  */
}


/* PROCEDURE PROCESS_DATA_1 processes line 1 of a data hit in TX format */

void process_data_1(char line[80],
                        double *lat,
                        double *lon,
                        double *class,
                        double *day,
                        double *pass_time)

{
 char  lat_str[22]="";
 char  lon_str[8]="";
 char  class_str[8]="";
 char  day_str[8]="";
 char  pass_time_str[8]="";
 char  lat_hem;
 char  lon_hem;
 char *end_str;

 /* extract parameters from input string */

 sscanf(line,"%*8c%6c%c%*c%7c%c",lat_str,&lat_hem,lon_str,&lon_hem);
```

```
sscanf(line,"%*27c%7s",class_str);
sscanf(line,"%*40c%3s",day_str);
sscanf(line,"%*44c%4s",pass_time_str); /**/


/* convert strings to numerics */

 *lat = strtod(lat_str, &end_str);
 *lon = strtod(lon_str, &end_str);
 *class = strtod(class_str, &end_str);
 *day = strtod(day_str, &end_str);
 *pass_time = strtod(pass_time_str, &end_str);

 *lat *= 2*(lat_hem == 'N')-1;
 *lon *= 2*(lon_hem == 'E')-1;


/* END procedure  PROCESS_DATA_1 */

}


/* PROCEDURE PROCESS_DATA_2 processes line 2 of a data hit in TX format */


void process_data_2(char line[80],
                        double mins[],
                        double maxs[],
                        double aves[],
                        double stddevs[]) /**/


{ char ch1_str[] = "12345678";
  char ch2_str[] = "12345678";
  char ch3_str[] = "12345678";
  char ch4_str[] = "12345678";
  char min_str[] = "  ", max_str[] = "  ",
       ave_str[] = "  ", stddev_str[] = "  ";
  char *end_str;



  sscanf(line,"%*9c%8c%*5c%8c%*5c%8c%*5c%8c",
         ch1_str, ch2_str, ch3_str, ch4_str);

  /* process channel 1   Lu683  */

  sscanf(ch1_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

  mins[1] = hextod(min_str);
  maxs[1] = hextod(max_str);
  aves[1] = hextod(ave_str);
  stddevs[1] = hextod(stddev_str);


  /* process channel 2   Lu670  */

  sscanf(ch2_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);
```

```
mins[2] = hextod(min_str);
maxs[2] = hextod(max_str);
aves[2] = hextod(ave_str);
stddevs[2] = hextod(stddev_str);

/* process channel 3   Lu555   */

sscanf(ch3_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

mins[3] = hextod(min_str);
maxs[3] = hextod(max_str);
aves[3] = hextod(ave_str);
stddevs[3] = hextod(stddev_str);

/* process channel 4   Lu510    */

sscanf(ch4_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

mins[4] = hextod(min_str);
maxs[4] = hextod(max_str);
aves[4] = hextod(ave_str);
stddevs[4] = hextod(stddev_str);


/* END procedure PROCESS_DATA_2 */

}


/* PROCEDURE PROCESS_DATA_3 processes line 3 of a data hit in TX format */


void process_data_3(char line[80],
                             double mins[],
                             double maxs[],
                             double aves[],
                             double stddevs[],
                             double *irrad_ave,
                             double *irrad_std) /**/


{ char ch1_str[] = "12345678";
  char ch2_str[] = "12345678";
  char ch3_str[] = "12345678";
  char ch4_str[] = "1234";
  char min_str[] = "  ", max_str[] = "  ",
       ave_str[] = "  ", stddev_str[] = "  ";
  char *end_str;


  sscanf(line,"%*9c%8c%*5c%8c%*5c%8c%*9c%4c",
         ch1_str, ch2_str, ch3_str, ch4_str);

  /* process channel 5   Lu490 */

  sscanf(ch1_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

  mins[5] = hextod(min_str);
```

```
maxs[5] = hextod(max_str);
aves[5] = hextod(ave_str);
stddevs[5] = hextod(stddev_str);

/* process channel 6   Lu443 */

sscanf(ch2_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

mins[6] = hextod(min_str);
maxs[6] = hextod(max_str);
aves[6] = hextod(ave_str);
stddevs[6] = hextod(stddev_str);

/* process channel 7   Lu412 */

sscanf(ch3_str,"%2c%2c%2c%2c",ave_str,min_str,max_str,stddev_str);

mins[7] = hextod(min_str);
maxs[7] = hextod(max_str);
aves[7] = hextod(ave_str);
stddevs[7] = hextod(stddev_str);

/* process irradiance  Ed490  */

sscanf(ch4_str,"%2c%2c",ave_str,stddev_str);

*irrad_ave = hextod(ave_str);
*irrad_std = hextod(stddev_str);

/* END procedure PROCESS_DATA_3 */
}

/* PROCEDURE PROCESS_DATA_4 processes line 4 of a data hit in TX format */

void process_data_4(char line[80],
                          double *temp,
                          double *tick) /**/

{ char ch1_str[] = "12";
  char ch2_str[] = "12"
  char *end_str;

  sscanf(line,"%*15c%2c%*11c%2c",ch1_str, ch2_str);

  /* process temperature */

  *temp = hextod(ch1_str);
  *tick = hextod(ch2_str);

/* END procedure PROCESS_DATA_4 */
```

```
/* FUNCTION  LOCATION_CLASS  reports accuracy of the platform location
                                                   in meters from the ARGO

double location_class(double class)

{
  double accuracy;

  accuracy =0.0;
  if (class == 1.0)
        accuracy = 1000.0;
  if (class == 2.0)
        accuracy = 350.0;
  if (class == 3.0)
        accuracy = 150.0;
  return accuracy;

/* END FUNCTION LOCATION_CLASS */

}

/* FUNCTION DEC_DATE  computes the decimal day from the day, hour, minute */

double dec_date(double day, double time)

{

 double hour, minutes, dec_day;

 hour = floor(time/100.0);
 minutes = time - hour * 100;
 dec_day = day + hour/24.0 + minutes/24.0/60.0;
 return dec_day;

/*  END FUNCTION DEC_DATE */

}

/* PROCEDURE PROCESS_HIT uses data extracted by the PROCESS_DATA_ procedures,
                                             applies calibration data and wr
                                             to the output file    */

void process_hit(FILE *PROCESS_data,
                 double day,
                 double time,
                 double lat,
                 double lon,
                 double class,
                 double aves[],
```

```c
                 double mins[],
                 double maxs[],
                 double stddevs[],
                 double irrad_ave,
                 double irrad_stddev,
                 double temp,
                 double tick)

/* process counts and produce physical units for each parameter */


double accuracy;
int channel,i;
double std_dev,battery;
double ave, min, max,tmp, minutes;
double date;

accuracy= location_class(class);
date=dec_date(day,time);


fprintf(PROCESS_data,"%s %9.4lf %6.3lf %6.3lf %5.0lf",platform,
               date,lat,lon,accuracy);

 /* apply calibrations to radiance channels */

for (channel=1; channel < 8; channel++)
{
  i=channel;

  tmp=aves[i];
  ave = cal_rad(tmp, channel);

  tmp=mins[i]; min = cal_rad(tmp, channel);

  tmp=maxs[i]; max = cal_rad(tmp, channel);

  tmp=stddevs[i];
  std_dev = stand_dev(tmp, aves[i]);
  std_dev = cal_rad(std_dev, channel);


  fprintf(PROCESS_data," %6.4lf", ave);

}

/* apply calibration to irradiance data */

ave = cal_irrad(irrad_ave);
std_dev = stand_dev(irrad_stddev,irrad_ave);
std_dev= cal_irrad(std_dev);

temp = cal_temp(temp);
minutes = get_tick(tick);
battery = get_volts(tick);

fprintf(PROCESS_data," %5.1lf %4.1lf %4.1lf %4.1lf\n",ave,
              temp,battery,minutes);
```

```
/* END PROCEDURE PROCESS_HIT  */



/*  PROCEDURE GET_CALIBRATION_VALS  reads in calibration data */

int get_calibration_vals()

{
 int error,i;
 char line[80]="";

 char first_char;


 i=0;
 error=0;

 while (!feof(CAL_data))
  {
        line[0] = 0;
        fgets(line,79,CAL_data);

        #if defined DEBUG
        printf("%s",line);
        #endif


        if (strlen(line) == 0 || line[0] == COMMENT) continue;
        sscanf(line,"%le %lf",scale+i,offset+i);

        #if defined DEBUG
        printf("%le %lf\n",scale[i],offset[i]);
        #endif

        i++;
        if ((i>8) && (line[0] != '\n'))
         {
          printf(" ERROR in calibration file - too many cal vals\n");
          error=-1;
         }
   }
  if (i<8)
    {
         printf(" ERROR in calibration file - too few cal vals\n");
         error=-1;
    }
  fclose(CAL_data);
  return error;
}

 /*  END PROCEDURE GET_CALIBRATION_VALS */
```

```
* PROCEDURE HIT */

nt hit(char line[80],
                                        char platform[6])

{
 char check[9]="";

   if (strstr(line,platform) == NULL) return 0;   /* if platform ID not in line

   sscanf(line,"%10s",check);   /* check first 10 chars for ID only */

   if (strstr(check, platform) == NULL)
        return 0;
   else
        return 1;

  }

/* END PROCEDURE PROCESS HIT */

/**   PROCEDURE MAIN   **/

main()

{
 FILE *ARGOS_data;     /* current ARGOS file */
 FILE *PROCESS_data;  /* output file */
 FILE *PROCESS_LIST;  /* list of files to process */

 char file_list[] = "infiles.lst";
 char infile_ext[] = ".ARG";
 char infile_name[13];
 char outfile_name[13];
 char cal_file_name[13];
 char line[80] = "";
 int line_no = 0;
 int data_hits = 0;

 double lat,lon,class,day,time;
 double mins[8], maxs[8], aves[8], stddevs[8];
 double irrad_ave, irrad_std;
 double temp;
 double tick_time;
 int i;


 /*  read in list of files to .ARG files to process */

 if ((PROCESS_LIST = fopen(file_list,"r")) == NULL)
```

```
{
 printf(" File INFILES.LST must exist in current directory\n");
 printf(" and must contain a list of files to process!\n");
 return -1;
}

/* get output file name for calibrated data */

puts(" ENTER output file name:");
scanf("%13s",outfile_name);
if ((PROCESS_data = fopen(outfile_name,"w")) == NULL)
 {
  printf(" ERROR creating %s \n",outfile_name);
  return -1;
 }


/* read in calibration file for current platform */

puts(" ENTER platform calibration file name:");
scanf("%13s",cal_file_name);

if ((CAL_data = fopen(cal_file_name,"r")) == NULL)
 {
  printf(" ERROR calibration file  %s not found\n",cal_file_name);
  printf(" Program Terminated \n");
  return -1;
 }


if (get_calibration_vals() == -1)
 {
  printf(" Program Terminated \n");
  return -1;   /*  read cal file */
 }


/* extract platform name from first 5 characters of cal file name */

sscanf(cal_file_name,"%5s",platform);


/* put text header on output file */

fprintf(PROCESS_data,"  ID      DAY     LAT    LON   ACCUR");
fprintf(PROCESS_data," Lu683");
fprintf(PROCESS_data,"  Lu670");
fprintf(PROCESS_data,"  Lu555");
fprintf(PROCESS_data,"  Lu510");
fprintf(PROCESS_data,"  Lu490");
fprintf(PROCESS_data,"  Lu443");
fprintf(PROCESS_data,"  Lu412");
fprintf(PROCESS_data,"  ED490 Temp Vbat Tick\n");


/* scan .ARG files and extract data for current platform */
```

```
while ( !feof(PROCESS_LIST) )
 {
        line[0]=0;
        infile_name[0]=0;
        fgets(line,80,PROCESS_LIST);

        if ( (line[0] == '\0') || (line[0] == ' ') || line[0] == '\n' ) continu
        sscanf(line,"%s",infile_name);


 /* scan current .ARG file for current platform */


if ((ARGOS_data = fopen(infile_name,"r")) != NULL)
{
 fgets(line,80,ARGOS_data);   /* read line from file */
 line_no++;
 while ( !feof(ARGOS_data) )
  {

        /* check current line for correct plaform ID */

        if (hit(line,platform)) /* process data */
         {

          data_hits++;
          printf(" processing hit %3.0i platform %s\n",data_hits,platform);
          process_data_1(line, &lat, &lon, &class, &day, &time);

          fgets(line,80,ARGOS_data);
          process_data_2(line, mins, maxs, aves, stddevs);
          line_no++;

          fgets(line,80,ARGOS_data);
          process_data_3(line, mins, maxs, aves, stddevs, &irrad_ave, &irrad_st
          line_no++;

          fgets(line,80,ARGOS_data);
          process_data_4(line, &temp, &tick_time);
          line_no++;

          process_hit(PROCESS_data,day,time,lat,lon,class,aves,mins,maxs,
                  stddevs,irrad_ave,irrad_std,temp,tick_time);
         }
        fgets(line,80,ARGOS_data);   /* read next line */
        line_no++;
  }
 line_no--;
 fclose(ARGOS_data);
 }
 else
        printf(" Error file: %s not found\n",infile_name);

 } /* end while processing */

 fclose(PROCESS_LIST);
 fclose(PROCESS_data);
 printf(" %i lines processed\n",line_no);
 printf(" %i data hits processed\n",data_hits);
 return 0;
```

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

APPENDIX 2

Notes on TX formatted data from the ARGOS Users Manual

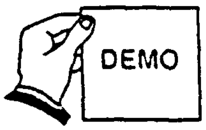
Telephone: (902) 492-4780 ***Facsimile: (902) 492-4781***

3.2.5 - DOWNLOADING RESULTS IN TX FORMAT

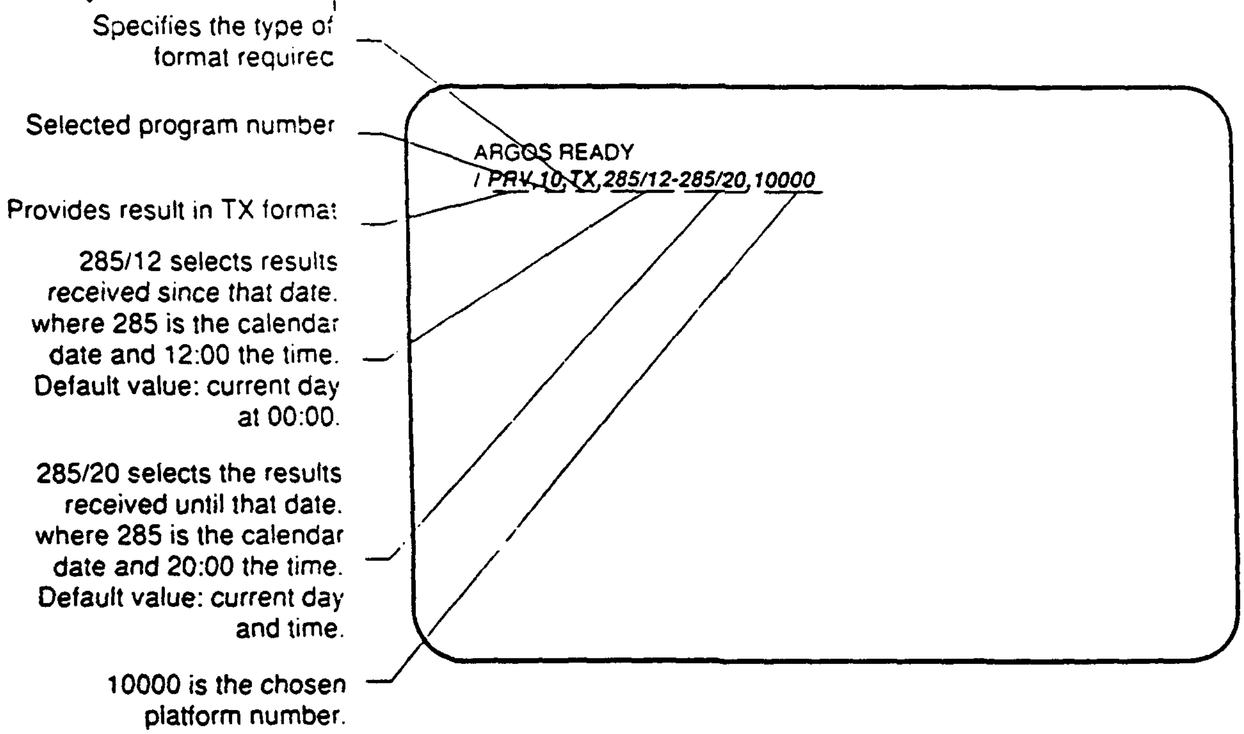
Command PRV lets you obtain the results, in the TX format, generated by one or more platforms in a program or programs of which you are the owner. The results for the current day and the four previous days can be provided.

Use command PRV to select the results you wish to download by entering the platform numbers and message reception dates.

COMMAND PRV

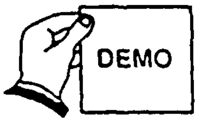


Enter the command as follows:



The above command was executed on September 13, 1987 (calendar day: 286). The selected dates, entered in the command (285/12 - 285/20), are obviously examples only and will depend on the date of your interrogation.

Press Carriage Return; the following results appear:




```
ARGOS READY
/ PRV,10,TX,285/12-285/20,10000

PROG 00010

10000  43.543N   1.401E 3              285/1340Z-285/1336
 (2)     0.10486E+4            00  0.20000E+2  0.32000E+2

10000  43.544N   1.399E 3              285/1521Z-285/1515
 (12)    0.99875E+3            32  0.20000E+2  0.31000E+2

10000  43.544N   1.399E 3              285/1701Z-285/1515
 (6)     0.99927E+3            32  0.20000E+2  0.30000E+2

ARGOS READY
/
```

These three sets of results represent the most significant messages received on the 285th day between 12:00 and 20:00. The structure of the three sets is identical.

Interpret the above results as follows:

- Location class
- Platform longitude in degrees and thousandths of a degree
- Selected program number
- Selected platform number
- Calendar day and time (HHMM) of last location in UTC.
- Calendar day and time (HHMM) of data collection in UTC.
- Compression index. Twelve identical messages were received.
- Platform latitude in degrees and thousandths of a degree
- Physical value for sensor 1 (998.75 hPa)
- Hexadecimal value for sensor 2
- Physical value for sensor 3 (20°C)
- Physical value for sensor 4 (31°C)

```
ARGOS READY
/ PRV,10,TX,285/12-285/20,10000

PROG 00010

10000  43.543N   1.401E 3              285/1340Z-285/1336
 (2)     0.10486E+4            00  0.20000E+2  0.32000E+2

10000  43.544N   1.399E 3              285/1521Z-285/1515
 (12)    0.99875E+3            32  0.20000E+2  0.31000E+2

10000  43.544N   1.399E 3              285/1701Z-285/1515
 (6)     0.99927E+3            32  0.20000E+2  0.30000E+2

ARGOS READY
/
```

To request on-line information on command PRV, enter "?,PRV" and a Carriage Return.
To abort the current command, enter "A".
To interrupt the current command, enter "I" (upper-case). To resume, press Carriage Return.

Command PRV can be shortened to "P".

The examples below show the options available under command PRV.

/PRV,10,TX,,
Downloads all messages processed for program 10 received on the current day since 00.00 UTC.

/PRV,10,TX, 285,
Downloads all messages processed for program 10 received between 00.00 UTC on day 285 and the current day. In practice, only the results of the four days prior to the current day are accessible.

/P,10,TX,285/18,
Downloads all messages processed for program 10 received between day 285 at 18.00 and the current day. In practice, only the results of the four days prior to the current day are accessible.

/P,,TX,285-286,
Downloads all messages processed (for all platforms user has access to) received between 00.00 on day 285 and 24.00 on day 286.

Satlantic Inc.

Richmond Terminal, Pier 9
3295 Barrington Street
Halifax, Nova Scotia
Canada B3K 5X8

APPENDIX 3

Using OCM_PRO4.C an example

Includes: sample data file SEP05.ARG
sample calibration file 04742A.CAL
sample calibration file 04745A.CAL
sample batch processing file INFILES.LST
sample of OCM_PRO4 executing on SEP05.ARG
sample output 4742_05.DAT
sample output 4745_05.DAT

Telephone: (902) 492-4780 ***Facsimile: (902) 492-4781***

September 06, 1992 15:48 hr
LOGIN AT 249/1841 LAST ACCESS AT 248/1345 UTC

ARGOS READY
/PRV,895,TX,248/13-249/18.
SYNTAX ERROR

ARGOS READY
/PRV,895,TX,248/13-249/18,
Prog 00895

```
04742  36.748N 121.865W   0          248/1353Z-248/1352
( 1)             00          35          21          10
                 11          16          06          14
                 7B          CF

04742  36.746N 121.864W   2          248/1510Z-248/1506
( 1)            13B         145     1000356     100035E
            200066E     200056A     2000464         268
                 7A          E1

04742  36.745N 121.863W   3          248/1649Z-248/1646
( 1)        4030679     4020573     C080F96     F0A139E
           191120AD    140E1AA7    120C17A3        12A3
                 7A          CD

04742  36.745N 121.863W   3          248/1824Z-248/1646
( 1)        6050883     504067D    110F14A2    167E003F
           FFFFFFFF    FFFFFFFF    FF76FFFF        FFFF
                 F1          C7

04742  36.747N 121.867W   2          248/2202Z-248/2158
( 1)        8060A8B     7050886    18121DAC    1D1622B2
           2F2237C1    281F2EBC    251E2EBA        21B6
                 7B          DE

04742  36.747N 121.866W   3          248/2340Z-248/2339
( 1)        7040B88     6040980    130B21A5    170E27AB
           24163BB9    1E1327B3    1B1225B0        1AAE
                 7B          C9

04742  36.747N 121.866W   3          249/0047Z-248/2339
( 1)        6030881     4020678     E06129B    100816A0
           1A0D22AE    140A1CA7    130919A5        13A6
                 7B          D4

04742  36.746N 121.867W   2          249/0230Z-249/0226
( 1)        100024D     100024E     2000461     2000566
            3000773     200056C     2000569         370
                 7B          C1

04742  36.749N 121.864W   2          249/0410Z-249/0407
```

```
( 1)        00          34          22          15
      FF55FFE7    FFFFDEC3    BFEF7BF9        EFC6
            21          39

04742  36.747N 121.863W   2        249/1023Z-249/1019
( 1)        00          34          21          0E
            0D          14          00          0F
            7B          C0

04742  36.745N 121.867W   2        249/1207Z-249/1202
( 1)        00          34          21          0F
            0C          12          00          0E
            7B          ED

04742  36.750N 121.868W   1        249/1342Z-249/134C
( 1)        00          34          1F          0D
            0F          14          00          11
            7B          D5

04742  36.745N 121.862W   2        249/1448Z-249/1445
( 1)        1D          3A         13A         13D
           24B         249         245          47
            7B          DC

04742  36.755N 121.848W   0        249/1620Z-249/1619
( 1)   3020572     302036B     7040988     9050B8D
       E09129C     E030F96     A071CE3        9C71
            C6          38

04744   0.165S 140.958W   2        248/1408Z-248/1406
( 1)        0A          00          00          12
            27          31          2F          00
            BA          D9

04744   0.165S 140.958W   2        248/1517Z-248/1406
( 1)        0A          00          00          12
            27          38          2E          00
            BA          E7

04744   0.165S 140.958W   2        248/1701Z-248/1406
( 1)       23E         242     2000565     4010A7C
       A021795     A031796     C031A9A         47D
            BA          DC

04744   0.165S 140.958W   2        248/1837Z-248/1406
( 1)   2010361     2010461     9050C8F    13091AA6
      2C1A38BF    30153DC2    3A204DC8        16AB
```

```
                  BA            C9

 04744   0.269S 141.187W   2              248/2329Z-248/2328
 ( 1)     2010365       200066A      F0A149E       1F1325B4
         492C62CF       513D69D2     673385DA          26BA
                  BD            D0

 04744   0.269S 141.187W   2              249/0114Z-248/2328
 ( 1)     2000360       2000362      A030E92       15071EA9
         2E133CC1       321740C3     411E55CC          19AF
                  BD            C7

 04744   0.288S 141.227W   2              249/0116Z-249/0109



 ( 1)     2000360       2000362      A030E92       15071EA9
         2E133CC1       321740C3     411E55CC          19AF
                  BD            C9

 04744   0.308S 141.291W   2              249/0400Z-249/0356
 ( 1)          0A            0E           24            241
          100045D       1000561      1000563             49
                  BC            C1

 04744   0.334S 141.330W   2              249/0539Z-249/0535
 ( 1)          0A            00           00             11
               2B            3C           30             00
               BC            EE

 04744   0.424S 141.486W   2              249/1218Z-249/1213
 ( 1)          0A            00           00             0D
               2C            37           31             00
               BB            EE

 04744   0.441S 141.513W   2              249/1358Z-249/1353
 ( 1)          0A            00           0A             0D
               2F            00           00             00
               00            00

 04745   4.429N 142.307W   2              248/1407Z-248/1404
 ( 1)          0A            0C           03             0A
               0A            25           25             00
               BA            D8

 04745   4.429N 142.307W   2              248/1517Z-248/1404
 ( 1)          0A            0A           03             0A
               0A            25           24             00
               BA            E4
```

```
  04745   4.446N 142.328W   2          248/1659Z-248/1655
  ( 1)        00            31        1000250       300066F
          802108C      A031394       C04169A           373
              BA            D6

  04745   4.461N 142.339W   2          248/1838Z-248/1835
  ( 1)    1000254      100025D       7010988       15051BA8
          340E44C5     3F1352CB      4F1A65D2          14A8
              BB            C4

  04745   4.495N 142.385W   2          248/2334Z-248/2330
  ( 1)    100065B      2010563       A050E91       1C0F26B2
          482865CF     5B3380D6      76429DDE          1FB4
              BE            CC

  04745   4.490N 142.383W   0          249/0115Z-248/2335
  ( 1)    1000254      201025F       9050B8E       190F1FAD
          3F2752CA     4E2C66D1      643D82D9          1AAF
              BF            F9

  04745   4.526N 142.395W   2          249/0116Z-249/0111
  ( 1)    1000254      201025F       9050B8E       190F1FAD




          3F2752CA     4E2C66D1      643D82D9          1AAF
              BF            FA

  04745   4.557N 142.425W   2          249/0401Z-249/0357
  ( 1)        0E           13A       100035F        400097B
          B021798      E031B9E       110423A5           585
              BE            F1

  04745   4.586N 142.431W   2          249/0541Z-249/0537
  ( 1)        0A            0D            02             0A
              0A            25            25             00
              BD            E1

  04745   4.645N 142.499W   2          249/1217Z-249/1211
  ( 1)        0A            0B            02             0A
              0A            25            25             00
              BC            D8

  04745   4.656N 142.522W   2          249/1355Z-249/1352
  ( 1)        0A            0A            01             0A
              0A            24            24             00
              BC            C6

ARGOS READY
/LO
```

\SCOTT\EXPERIME.NTS\JGOFS\BUOY\04742A.CAL Page: 1 1
esday July 28, 1993 07:28:02 pm

bration File - OCM Shiplaunch ID4742 OCR-100 S/N 007 ED-100 S/N 010
lid for Data collected between Aug 10/92 and Oct/92
bration Date Aug 10/92 - Lamp S721
r Lu683, Lu670, Lu555, Lu510, Lu490, Lu443, Lu412, Ed490
e equation CalRad = scale * (counts - offset)
5e-3 0
-3 0
-3 0
9e-3 0
-3 0
-3 0
0e-3 0
8 0

AUG31.ARG
SEP01.ARG
SEP02.ARG
SEP03.ARG
SEP04.ARG
SEP05.ARG
SEP06.ARG
SEP08.ARG
SEP09.ARG
SEP10.ARG
SEP11.ARG
SEP14.ARG
SEP16.ARG
SEP18.ARG
SEP19.ARG
SEP21.ARG
SEPT22.ARG
SEP24.ARG

```
C:\USR\SCOTT\EXPERIME.NTS\JGOFS\BUOY>OCM_PRO4
ENTER output file name:
4742_05.DAT
ENTER platform calibration file name:
04742A.CAL
processing hit   1 platform 04742
processing hit   2 platform 04742
processing hit   3 platform 04742
processing hit   4 platform 04742
processing hit   5 platform 04742
processing hit   6 platform 04742
processing hit   7 platform 04742
processing hit   8 platform 04742
processing hit   9 platform 04742
processing hit  10 platform 04742
processing hit  11 platform 04742
processing hit  12 platform 04742
processing hit  13 platform 04742
processing hit  14 platform 04742
232 lines processed
14 data hits processed
```

```
C:\USR\SCOTT\EXPERIME.NTS\JGOFS\BUOY>OCM_PRO4
ENTER output file name:
4745_05.DAT
ENTER platform calibration file name:
04745A.CAL
processing hit   1 platform 04745
processing hit   2 platform 04745
processing hit   3 platform 04745
processing hit   4 platform 04745
processing hit   5 platform 04745
processing hit   6 platform 04745
processing hit   7 platform 04745
processing hit   8 platform 04745
processing hit   9 platform 04745
processing hit  10 platform 04745
processing hit  11 platform 04745
232 lines processed
11 data hits processed
```

```
C:\USR\SCOTT\EXPERIME.NTS\JGOFS\BUOY\4742_05.DAT                    Page:   1
Wednesday  July 28, 1993   07:28:03 pm

 ID        DAY      LAT     LON    ACCUR Lu683  Lu670  Lu555  Lu510  Lu490  Lu443  L
u412  ED490 Temp Vbat Tick
04742  248.5785 36.748 -121.865      0 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000    0.0 14.7 15.0 15.0
04742  248.6319 36.746 -121.864    350 0.0000 0.0000 0.0286 0.0268 0.0365 0.0289
0.0271    9.2 14.5 15.0 33.0
04742  248.7007 36.745 -121.863    150 0.0362 0.0437 0.3432 0.4024 0.4560 0.2891
0.2440   82.8 14.5 15.0 13.0
04742  248.7667 36.745 -121.863    150 0.0543 0.0547 0.4862 0.5902 4.6516 3.6862
.4570 1172.5 33.6 15.0  7.0
04742  248.9181 36.747 -121.867    350 0.0724 0.0766 0.6864 0.7780 0.8574 0.5782
0.5016 151.7 14.7 15.0 30.0
04742  248.9861 36.747 -121.866    150 0.0634 0.0656 0.5434 0.6171 0.6567 0.4337
.3660 119.5 14.7 15.0  9.0
04742  249.0326 36.747 -121.866    150 0.0543 0.0437 0.4004 0.4293 0.4743 0.2891
0.2576  87.4 14.7 15.0 20.0
04742  249.1042 36.746 -121.867    350 0.0091 0.0109 0.0572 0.0537 0.0547 0.0289
0.0271   13.8 14.7 15.0  1.0
04742  249.1736 36.749 -121.864    350 0.0000 0.0000 0.0000 0.0000 4.6516 3.6862
2.5893 1098.9  0.3 10.5 57.0
04742  249.4326 36.747 -121.863    350 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000    0.0 14.7 15.0  0.0
04742  249.5049 36.745 -121.867    350 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000    0.0 14.7 15.0 45.0
04742  249.5708 36.750 -121.868   1000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000    0.0 14.7 15.0 21.0
04742  249.6167 36.745 -121.862    350 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000    0.0 14.7 15.0 28.0
04742  249.6806 36.755 -121.848      0 0.0272 0.0328 0.2002 0.2415 0.2554 0.2024
0.1356 717.3 26.7 10.5 56.0
```

ID	DAY	LAT	LON	ACCUR	Lu683	Lu670	Lu555	Lu510	Lu490	Lu443	Lu412	ED490	Temp	Vbat	Tick
04745	248.5882	4.429	-142.307	350	0.0100	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	24.8	15.0	24.0
04745	248.6368	4.429	-142.307	350	0.0100	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	24.8	15.0	36.0
04745	248.7076	4.446	-142.328	350	0.0100	0.0000	0.0269	0.0771	0.1439	0.1385	0.1503	14.9	24.8	15.0	22.0
04745	248.7764	4.461	-142.339	350	0.0199	0.0107	0.1882	0.5398	0.9350	0.8725	0.9896	99.6	24.9	15.0	4.0
04745	248.9819	4.495	-142.385	350	0.0199	0.0214	0.2689	0.7197	1.2947	1.2603	1.4781	154.4	25.4	15.0	12.0
04745	249.0521	4.490	-142.383	0	0.0199	0.0214	0.2420	0.6426	1.1328	1.0803	1.2526	129.5	25.6	15.0	57.0
04745	249.0528	4.526	-142.395	350	0.0199	0.0214	0.2420	0.6426	0.3237	0.2493	0.2255	89.6	-2.1	10.5	18.0
04745	249.1674	4.557	-142.425	350	0.0100	0.0000	0.0269	0.1028	0.1978	0.1939	0.2129	24.9	25.4	15.0	49.0
04745	249.2368	4.586	-142.431	350	0.0100	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	25.2	15.0	33.0
04745	249.5118	4.645	-142.499	350	0.0100	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	25.1	15.0	24.0
04745	249.5799	4.656	-142.522	350	0.0100	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0	25.1	15.0	6.0